EXHIBIT Y

TO DECLARATION OF S. MERRILL WEISS IN SUPPORT OF PLAINTIFF ACACIA MEDIA TECHNOLOGIES CORPORATION'S MEMORANDUM OF POINTS AND AUTHORITIES IN OPPOSITION TO ROUND 3 DEFENDANTS' MOTION FOR SUMMARY JUDGMENT OF INVALIDITY UNDER 35 U.S.C. § 112 OF THE '992, '863, AND '702 PATENTS; AND SATELLITE DEFENDANTS' MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF THE '992, '863, AND '720 PATENTS

Hovsepian v. Apple, Inc.

Doc. 325 Att. 25

Dockets.Justia.com

Vector Quantization in Speech Coding

JOHN MAKHOUL, FELLOW, IEEE, SALIM ROUCOS, MEMBER, IEEE, AND
HERBERT GISH, MEMBER, IEEE

Invited Paper

***Quantization**, the process of approximating continuous-amplitude signals by digital (discrete-amplitude) signals, is an important aspect of **data compression** or **coding**, the field concerned with the reduction of the number of bits necessary to transmit or store analog data, subject to a distortion or fidelity criterion. The independent quantization of each signal value or parameter is termed **scalar** quantization, while the joint quantization of a block of parameters is termed block or **vector** quantization. This tutorial review presents the basic concepts employed in vector quantization and gives a realistic assessment of its benefits and costs when compared to scalar quantization. Vector quantization is presented as a process of redundancy removal that makes effective use of four interrelated properties of vector parameters: linear dependency (correlation), nonlinear dependency, shape of the probability density function (pdf), and vector dimensionality itself. In contrast, scalar quantization can utilize effectively only linear dependency and pdf shape. The basic concepts are illustrated by means of simple examples and the theoretical limits of vector quantizer performance are reviewed, based on results from rate-distortion theory. Practical issues relating to quantizer design, implementation, and performance in actual applications are explored. While many of the methods presented are quite general and can be used for the coding of arbitrary signals, this paper focuses primarily on the coding of speech signals and parameters.*

I. Introduction

Current projections for world-wide communications in the 1990s and beyond, point to a proliferation of digital transmission as a dominant means of communication for voice and data. Digital transmission is expected to provide flexibility, reliability, and cost effectiveness, with the added potential for communication privacy and security through encryption. The costs of digital storage and transmission media are generally proportional to the amount of digital data that can be stored or transmitted. While the cost of such media decreases every year, the demand for their use increases at an even higher rate. Therefore, there is a continuing need to minimize the number of bits necessary to transmit signals while maintaining acceptable signal fidelity or quality. The branch of electrical engineering that deals with the latter problem is termed *data compression* or *coding*. When applied to speech, it is known as *speech compression* or *speech coding*.

The conversion of an analog (continuous-time, continuous-amplitude) source into a digital (discrete-time, discrete-amplitude) source, consists of two parts: *sampling* and *quantization*. Sampling converts a continuous-time signal into a discrete-time signal by measuring the signal value at regular intervals of time. Quantization converts a continuous-amplitude signal into one of a set of discrete amplitudes, thus resulting in a discrete-amplitude signal that is different from the continuous-amplitude signal by the quantization error or noise. In this paper, we shall assume that our signals are adequately sampled (see, for example, [107]) so that the only loss in fidelity is attributable to quantization.

When each of a set of parameters (or a sequence of signal values) is quantized separately, the process is known as *scalar quantization*. When the set of parameters is quantized jointly as a single vector, the process is known as *vector quantization* (also known as block quantization or pattern-matching quantization). We shall often abbreviate vector quantization in this paper as VQ.

A. Purpose and Scope

The main purpose of this paper is to present the reader with information that can be used in making a realistic assessment of the benefits and costs of vector quantization relative to scalar quantization, especially in speech coding applications. The emphasis is on the exposition of basic principles rather than the elaboration of various techniques and their variations for which references to the literature are provided. Vector quantization is presented as a process of redundancy removal that makes effective use of four interrelated properties of vector parameters: linear dependency (correlation), nonlinear dependency, shape of the probability density function (pdf), and vector dimensionality itself. We shall see that linear dependency and pdf shape can be employed quite effectively with scalar quantization while the other two properties cannot. Nonlinear dependency plays a significant role in the quantization of speech spectral parameters, while dimensionality is important for waveform quantization. Because of the relatively large cost of vector quantization (generally exponential in the number of dimensions and the number of bits per dimension), given today's computation and storage technologies, the major benefits of vector quantization are

Manuscript received January 15, 1985; revised July 2, 1985.
The authors are with BBN Laboratories Inc., Cambridge, MA 02238, USA.

0018-9219/85/1100-1551$01.00 ©1985 IEEE

Exhibit Y Page 459

realized largely at transmission rates of about 1 bit per parameter or less, which is exactly the range where the performance of scalar quantizers degrades sharply. While the concepts presented are quite general, we shall focus in this paper on the low-rate coding of speech (below 8 kbits/s) as an application.

Vector quantization for the purpose of speech coding was used by Dudley [31] in the 1950s and Smith [127] in the 1960s. However, it was not until the introduction of linear predictive coding (LPC) [8], [71], [86], [90] to speech coding that VQ has had significant activity, starting with the work of Kang and Coulter [77], but spurred on mainly by the work of Buzo *et al.* [18], [81]. Until recently, the main purpose for the use of VQ in speech coding has been to reduce the transmission rate of 2400-bit/s vocoders (voice coders) to operate at much lower rates while maintaining acceptable speech intelligibility and quality. Speech coding at very low rates, in the range of 200–800 bits/s, has attracted substantial interest [18], [32], [53], [56], [76], [77], [98], [100], [111], [119], [137], [138] for use in both government and commercial applications. At such low rates, it is important to maximize the cost effectiveness of every bit that is transmitted. Vector quantization has been instrumental in retaining sufficient speech intelligibility to make such systems of actual utility. Today, very-low-rate coding of speech remains one of the major successful applications of VQ. More recently, a mushrooming research activity in the application of VQ to speech waveform coding at somewhat higher data rates has been taking place. While much of the activity has focused on the 8–16-kbit/s range [1], [24], [25], [34], [37], [42], [49], [50], [56], [63], [83], [109], [124], some work has started at data rates below 8 kbits/s [7], [117], which points to exciting possibilities for high-quality speech coding at low rates.

We should point out that, in a sense, VQ has been used regularly and effectively in pattern-recognition type of speech applications, such as in speech and speaker recognition (see, for example, [27], [80], [95], [104], [106], [118], [126]). After all, the VQ problem is part of the general pattern-recognition problem of the classification of data into a discrete number of categories that optimize some fidelity criterion. Indeed, in the design of vector quantizers, one often employs well-known techniques from pattern recognition. However, the basic theory underlying VQ stems from information theory and has wider implications for the transmission of information.

The theoretical foundations of data compression and vector quantization lie in a branch of information theory known as rate-distortion theory, originally set forth by Shannon [122]. Also, most of the theoretical developments since Shannon have taken place as part of the information theory discipline. Of particular relevance is the book by Berger [14] on rate-distortion theory, as well as other information theory texts, such as [46], [92]. Because a full development of vector quantization theory would be highly mathematical, we have chosen in this paper to concentrate on presenting the basic notions and the major results with just enough mathematics that would allow us to be complete without being obscure, we hope.

For further reading, we list a few key references which also contain other references to the literature. The books of collected papers edited by Jayant [73] and Davisson and Gray [26] are devoted to data compression and cover aspects of speech compression. Two relatively recent special issues, the IEEE TRANSACTIONS ON INFORMATION THEORY of March 1982 and the IEEE TRANSACTIONS ON COMMUNICATIONS of April 1982, are devoted to quantization and to speech coding, respectively. The most comprehensive treatment of waveform coding, with applications to speech and video, is the recent book by Jayant and Noll [74]. The review articles by Gold [52], Flanagan *et al.* [39], and Makhoul [87] cover various aspects of speech coding, while the review articles by Gersho and Cuperman [49] and Gray [56] describe some recent work in vector quantization.

B. *Paper Outline*

In Section II we present the basic VQ problem and several distortion measures that are utilized, along with the basic system design and associated computational and storage costs. The section ends with a VQ model that is introduced, with examples, to aid the reader in visualizing the different processes at work in VQ and in assessing the relative merits of vector and scalar quantization. The remainder of the paper can be viewed as an elaboration of the basic model, supported by theoretical and practical results. Section III contains some of the major theoretical results known about VQ performance from rate-distortion theory. Section IV is devoted to the design of scalar quantizers for vector sources; it includes a comparison between scalar and vector quantization for low-rate speech coding. In Section V we present important practical considerations for vector quantizer design, including methods for reducing computational and storage costs at some loss in performance, and issues of robustness in terms of expected differences between design performance and operational performance. One can take advantage of long-term time-related signal dependencies to reduce the bit rate further without sacrificing signal fidelity; several time-dependent VQ methods are summarized in Section VI. The main paper ends in Section VII with a brief discussion of VQ in speech waveform coding and an outlook to the future.

C. *Speech Coding*

Before we start the main presentation, we describe the main components of a general speech coding system and two paradigms that we shall use in this paper as a basis for our examples from speech coding.

Fig. 1 shows the basic components of a data compression system appropriate for speech coding. The first component analyzes the discrete-time signal $s(n)$[1] and extracts a vector of unquantized parameters $\boldsymbol{x}(n)$. The set of parameters $\boldsymbol{x}(n)$ is quantized into the vector $\boldsymbol{y}(n)$, which is then encoded into a sequence of bits $\boldsymbol{c}(n)$ and transmitted through the transmission channel or stored in some storage medium. (The quantizer includes any prediction and feedback loops that are an integral part of the quantization process.) In general, the output of the channel $\boldsymbol{c}'(n)$ will be different from $\boldsymbol{c}(n)$ if there are channel errors. At the receiver, the decoder converts the sequence of bits $\boldsymbol{c}'(n)$ into parameter values $\boldsymbol{y}'(n)$, which are then used as input

[1]The dependence of the discrete-time signals on the sampling period will be assumed but not shown explicitly.

Exhibit Y Page 460

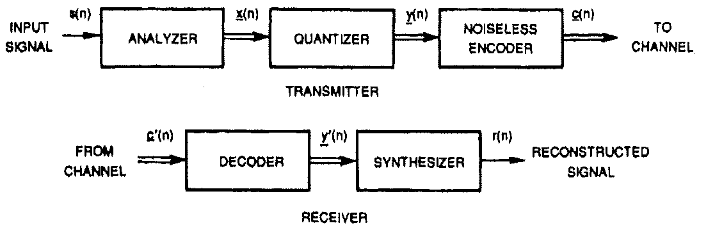

Fig. 1. Basic components of a data compression system for speech coding.

to the synthesizer. The output $r(n)$ is the reconstructed signal which will be an approximation to the input signal $s(n)$.

If there are no channel errors, then $c'(n) = c(n)$ and $y'(n) = y(n)$. The subject of channel errors is important, but will be treated only briefly in this paper. Unless otherwise noted, we shall assume no channel errors.

The nature of the synthesizer in Fig. 1 determines the type of voice coder and dictates the type of analysis to be performed. Fig. 2 shows an example of a synthesis model that is in general use. The model has two major components: an excitation (or source) and a spectral shaping filter. Having chosen a particular synthesis model, any reduction in transmission rate is accomplished by the quantizer and the encoder in Fig. 1. The encoder[2] is assumed to be noiseless, i.e., it does not introduce any additional noise or loss in fidelity. It assigns bits to $y(n)$ in such a way as to minimize the transmission rate, without any loss in fidelity, and may include additional bits to protect the transmitted bit stream against channel errors.

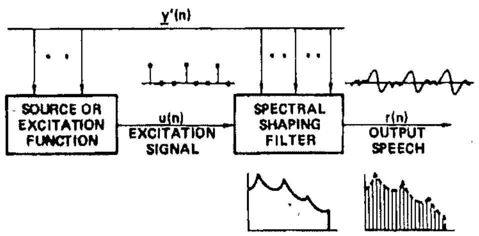

Fig. 2. Major components of the synthesizer in many speech coding systems.

The distortion in the output $r(n)$ relative to the input $s(n)$ may be the result of two processes: modeling and quantization. The modeling effected by the analysis/synthesis system in Fig. 1 may introduce a certain amount of distortion, even in the absence of any quantization (see, for example, the pitch-excited model described below). In this paper, we shall focus only on the distortion caused by the quantization process.

The speech coding task then is to design a system that minimizes the transmission rate while maintaining a certain speech quality, or conversely to maximize speech quality (minimize distortion) for a given transmission rate, subject to certain system cost constraints. For a given choice of analysis/synthesis system, the distortion and transmission rates are determined by the quantizer and the encoder.

[2]The terms "encoder" and "coding" are often used to refer to the whole compression process, as in speech coding. The noiseless encoder in Fig. 1 refers to a very specific part of the compression process. It is hoped that it will be clear from the context which of the two usages is meant.

Speech coding paradigms: We shall employ two basic paradigms as representative of the applications in speech coding, a low-rate coding paradigm and a medium-rate paradigm. The terms low-rate (or narrow-band) and medium-rate (or medium-band) have been used in the literature to denote a wide variety of systems. We shall use the term *medium-rate* for systems operating in the range 8–16 kbits/s, and *low-rate* for systems operating below that range, typically at or below 2400 bits/s. The term *very-low-rate* is often used to denote low-rate systems operating below about 1000 bits/s. Below, we give the basic paradigms that are used in this paper as examples of current systems that operate in the two main ranges.

In our paradigms, the synthesis model shown in Fig. 2 is based on a short-term spectral analysis of speech, where the speech signal is modeled as the output of an all-pole spectral shaping filter

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + \sum_{k=1}^{N} a(k) z^{-k}} \tag{1}$$

which is excited by a source with a flat spectral envelope. This is the well-known linear-predictive coding (LPC) model for speech. The gain G and the predictor coefficients $\{a(k), 1 \leqslant k \leqslant N\}$ are computed on a short-term basis over a *frame* of about 20–30 ms in which the speech signal can be considered to be approximately stationary. The coefficients are obtained as a result of minimizing the energy of the *prediction residual* obtained by filtering the input signal $s(n)$ through the all-zero filter $A(z)$. Ideally, the *excitation* signal $u(n)$ in Fig. 2 should match the residual signal so that the reconstructed signal $r(n)$ will match the input $s(n)$. In many medium-rate systems, the excitation used is exactly a quantized version of the prediction residual. This is true of many predictive waveform coding systems such as adaptive predictive coding (APC) [10], [89] and certain implementations of adaptive transform coding (ATC) [16]. We shall refer to such systems as *residual-excited* systems. (In Section VII, we shall include another filter that models the periodicity in the speech signal.)

At low rates, without using VQ it becomes necessary to have a simple model of the residual to maintain adequate speech intelligibility and quality. The most popular model is the *pitch-excited* model, where the speech in each frame is declared as either *voiced* or *unvoiced*. (Vowels and nasals are examples of voiced sounds, while consonants such as p, t, k, f, s, are unvoiced.) If a voiced determination is made, i.e., the sound is quasi-periodic at that point, the *pitch*, or fundamental frequency is measured and transmitted as well.

Exhibit Y Page 461

At the receiver, voiced sounds are synthesized by exciting the spectral shaping filter by a sequence of pulses separated by the period (Fig. 2 depicts that situation). For unvoiced sounds, a white random noise source is used as excitation. In either case, the gain G of the filter $H(z)$ is set such that the short-term energy in the output is equal to that of the input speech. Note that the pitch-excited model causes a certain amount of modeling distortion in the output, which can be heard even with no quantization of the model parameters.

II. Vector Quantization

This section begins with a formulation of the vector quantization problem, followed by a discussion of the more common distortion measures that are employed. Next we present the basic VQ system design and its associated computational and storage costs. The section ends with the introduction of a VQ model that is aimed at giving the reader a view of the various processes at work in vector quantization.

A. *Problem Formulation*

We assume that $\boldsymbol{x} = [x_1\ x_2 \cdots x_N]^T$ is an N-dimensional vector whose components $\{x_k, 1 \leqslant k \leqslant N\}$ are real-valued, continuous-amplitude random variables. (The superscript T denotes transpose.) In vector quantization, the vector $\boldsymbol{x}$ is mapped onto another real-valued, discrete-amplitude, N-dimensional vector $\boldsymbol{y}$. We say that $\boldsymbol{x}$ is quantized as $\boldsymbol{y}$, and $\boldsymbol{y}$ is the quantized value of $\boldsymbol{x}$. We write

$$\boldsymbol{y} = q(\boldsymbol{x})$$

where $q(\cdot)$ is the quantization operator. $\boldsymbol{y}$ is also called the *reconstruction vector* or the *output vector* corresponding to $\boldsymbol{x}$. Typically, $\boldsymbol{y}$ takes on one of a finite set of values $Y = \{\boldsymbol{y}_i, 1 \leqslant i \leqslant L\}$, where $\boldsymbol{y}_i = [y_{i1}\ y_{i2} \cdots y_{iN}]^T$. The set Y is referred to as the reconstruction *codebook*, or simply the codebook, L is the *size* of the codebook, and $\{\boldsymbol{y}_i\}$ are the set of *code vectors*. The vectors $\boldsymbol{y}_i$ are also known in the pattern-recognition literature as the reference patterns or *templates*. The size L of the codebook is also called the *number of levels*, a term borrowed from scalar quantization terminology. Thus one talks about an L-level codebook or L-level quantizer. To design such a codebook, we partition the N-dimensional space of the random vector $\boldsymbol{x}$ into L regions or *cells* $\{C_i, 1 \leqslant i \leqslant L\}$ and associate with each cell C_i a vector $\boldsymbol{y}_i$. The quantizer then assigns the code vector $\boldsymbol{y}_i$ if $\boldsymbol{x}$ is in C_i

$$q(\boldsymbol{x}) = \boldsymbol{y}_i, \qquad \text{if } \boldsymbol{x} \in C_i. \tag{2}$$

The codebook design process is also known as *training* or *populating the codebook*. A method for designing the codebook will be presented in Section II-C.

Fig. 3 shows an example of a partitioning of two-dimensional space ($N = 2$) for the purpose of vector quantization. The region enclosed by the bold lines is the cell C_i. Any input vector $\boldsymbol{x}$ that lies in the cell C_i is quantized as $\boldsymbol{y}_i$. The positions of the code vectors corresponding to the other cells are shown by dots. The total number of code vectors in the example of Fig. 3 is $L = 18$.

For $N = 1$, vector quantization reduces to scalar quantization. Fig. 4 shows an example of a partitioning of the real line for scalar quantization. The code values (output or reconstruction levels) are shown by dots. Here, also, any input value x that lies in the interval C_i is quantized as y_i. The number of levels in Fig. 4 is $L = 10$. Scalar quantization has the very special property that while cells may have different sizes, they all have the *same shape*, namely, they are all intervals on the real line. In comparison, note how in Fig. 3 the cells in two dimensions actually have different shapes. This freedom of having various cell shapes in multidimensional space gives vector quantization an advantage over scalar quantization, as we shall see below.

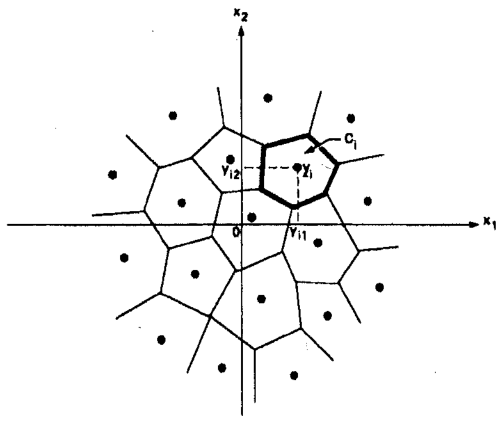

Fig. 3. Partitioning of two-dimensional space ($N = 2$) into $L = 18$ cells. All input vectors in cell C_i will be quantized as the code vector $\boldsymbol{y}_i$. The shapes of the various cells can be very different.

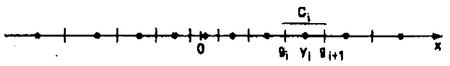

Fig. 4. Partitioning of the real line into $L = 10$ cells or intervals for scalar quantization ($N = 1$).

When $\boldsymbol{x}$ is quantized as $\boldsymbol{y}$, a quantization error results and a *distortion measure* $d(\boldsymbol{x}, \boldsymbol{y})$ can be defined between $\boldsymbol{x}$ and $\boldsymbol{y}$. $d(\boldsymbol{x}, \boldsymbol{y})$ is also known as a dissimilarity measure or distance measure. As the vectors $\boldsymbol{y}(n)$ at different times n are transmitted, one can define an overall average distortion

$$D = \lim_{M \to \infty} \frac{1}{M} \sum_{n=1}^{M} d[\boldsymbol{x}(n), \boldsymbol{y}(n)]. \tag{3}$$

If the vector process $\boldsymbol{x}(n)$ is stationary and ergodic,[3] the sample average in (3) tends in the limit to the expectation

$$\begin{aligned} D &= \mathscr{E}[d(\boldsymbol{x}, \boldsymbol{y})] \\ &= \sum_{i=1}^{L} P(\boldsymbol{x} \in C_i) \mathscr{E}[d(\boldsymbol{x}, \boldsymbol{y}_i) | \boldsymbol{x} \in C_i] \\ &= \sum_{i=1}^{L} P(\boldsymbol{x} \in C_i) \int_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{y}_i) p(\boldsymbol{x})\, d\boldsymbol{x} \end{aligned} \tag{4}$$

where $P(\boldsymbol{x} \in C_i)$ is the discrete probability that $\boldsymbol{x}$ is in C_i, $p(\boldsymbol{x})$ is the multidimensional probability density function (pdf) of $\boldsymbol{x}$, and the integral is taken over all components of the vector $\boldsymbol{x}$.

For purposes of transmission, each vector $\boldsymbol{y}_i$ is encoded

[3] Ergodicity allows us to substitute sample (or time) averages for ensemble averages [28].

Exhibit Y Page 462

into a codeword of binary digits (bits) c_i of length B_i bits. In general, the different codewords will have different lengths. The transmission rate T is then given by

$$T = BF_c \text{ bits/s} \tag{5}$$

where

$$B = \lim_{M\to\infty} \frac{1}{M} \sum_{n=1}^{M} B(n) \qquad \text{bits/vector} \tag{6}$$

is the average codeword length, $B(n)$ is the number of bits used to code the vector $x(n)$ at time n, and F_c is the number of codewords transmitted per second. It will also be useful to define the average number of bits per parameter or per dimension[4]

$$R = \frac{B}{N} \text{ bits/dimension.} \tag{7}$$

For a codebook of size L, the maximum number of bits needed to code each vector is

$$B_{\max} = \log_2 L. \tag{8}$$

In designing a data compression system, one attempts to design the quantizer such that the distortion in the output is minimized for a given transmission rate. One major decision in designing a quantizer is what distortion measure to use. This is discussed next.

B. Distortion Measures

To be useful, a distortion measure must be tractable, so that one can analyze it and compute it, and be subjectively relevant, so that differences in distortion values can be used as indicating similar differences in speech quality. Most distortion measures in use today are certainly tractable and, to some extent, subjectively relevant. However, many researchers have experienced the frustration which accompanies their discovery that a few decibels of decrease in the distortion is quite perceivable by the ear in one situation but not in another. The careful researcher has learned that, while objective distortion measures are necessary and useful tools in the design of speech coding systems, periodic subjective quality testing is indispensable to making an informed decision on directions for improving system performance. Below we list some of the major distortion measures in use today.

1) Mean-Square Error: By far the most common distortion measure is the mean-square error (mse)

$$d_2(x, y) = \frac{1}{N}(x - y)^T(x - y) = \frac{1}{N}\sum_{k=1}^{N}(x_k - y_k)^2 \tag{9}$$

where the distortion is defined per dimension. The popularity of the mse lies mainly in its simplicity and mathematical tractability. A more general distortion measure based on the L_r norm is defined by

$$d_r(x, y) = \frac{1}{N}\sum_{k=1}^{N}|x_k - y_k|^r. \tag{10}$$

[4]Note that R is the bit rate per dimension, B is the bit rate per vector, and T is the bit rate per second. We shall often use the generic term *bit rate* to refer to any or all of the three meanings. We trust the intended meaning will be clear from the context.

Note that (10) is equal to (9) for $r = 2$. The two other most popular values of r are $r = 1$ and $r = \infty$. d_1 represents the average absolute error and d_∞ tends towards the maximum error. In fact, one can show that

$$\lim_{r\to\infty} [d_r(x, y)]^{1/r} = \max\{|x_k - y_k|, 1 \leqslant k \leqslant N\}. \tag{11}$$

Minimizing D for $r = \infty$ would be equivalent to minimizing the maximum quantization error.

For speech coding, d_2 has been the most popular distortion measure, with d_1 and d_∞ being used occasionally.

2) Weighted Mean-Square Error: In the mse d_2 we assumed that the distortions contributed by quantizing the different parameters $\{x_k\}$ were weighted equally. In general, unequal weights can be introduced to render certain contributions to the distortion more important than others. A general weighted mse is then defined by

$$d_W(x, y) = (x - y)^T W(x - y) \tag{12}$$

where W is a positive-definite weighting matrix. $W = N^{-1}I$, where I is the identity matrix, results in $d_W = d_2$.

One choice for W that is popular in many pattern classification applications is $W = \Gamma^{-1}$, where Γ is the covariance matrix of the random vector x

$$\Gamma = E\left[(x - \bar{x})(x - \bar{x})^T\right], \qquad \bar{x} = \mathscr{E}[x]. \tag{13}$$

In this case, d_W reduces to

$$d_W(x, y) = (x - y)^T\Gamma^{-1}(x - y) \tag{14}$$

and is known as the Mahalanobis distance [85].

If, in addition to being positive definite, the weighting matrix W is symmetric (as in the Mahalanobis distance), one can factor W as follows:

$$W = P^TP. \tag{15}$$

The vectors x and y can be transformed into a new set of vectors $\tilde{x}$ and $\tilde{y}$

$$\tilde{x} = Px \qquad \tilde{y} = Py \tag{16}$$

and

$$\begin{aligned} d_W(x, y) &= (Px - Py)^T(Px - Py) \\ &= (\tilde{x} - \tilde{y})^T(\tilde{x} - \tilde{y}) \\ &= d_2(\tilde{x}, \tilde{y}). \end{aligned} \tag{17}$$

Thus the weighted mse between the original vectors is equal to the mse between the transformed vectors. Therefore, for computational purposes, it may be advantageous to perform the transformation in (16) on all the data before vector quantization is performed.

3) Linear Prediction Distortion Measures: In LPC analysis, the predictor coefficients $\{a(k)\}$ are obtained as a result of minimizing the energy of the prediction residual. One can show [86] that the solution for the optimum $A(z)$ in (1) is unique and is computed from the set of simultaneous linear equations

$$\sum_{k=1}^{N} a(k)\phi(i - k) = -\phi(i), \qquad 1 \leqslant i \leqslant N \tag{18}$$

where $\{\phi(i), 0 \leqslant i \leqslant N\}$ are the short-term autocorrelation coefficients of the speech signal over a single frame. (For a

Exhibit 4 Page 463

speech signal band-limited to 5 kHz, for example, the number of coefficients N is typically set to 12–14.) The gain G of the filter $H(z)$ is set so that when excited by a unit-variance source the output energy will be equal to $\phi(0)$. That can be accomplished by setting

$$G^2 = \phi(0) + \sum_{k=1}^{N} a(k)\phi(k) \tag{19}$$

which is equal to the minimum residual energy. The parameters of the filter $H(z)$ are computed every frame, quantized, and transmitted.

The gain G is usually quantized on a logarithmic scale and transmitted separately. (Some work has been done in joint quantization of the gain and the LPC parameters [108].) Because the quantization of predictor coefficients can lead to instability of the resulting all-pole filter, they are usually transformed to another set of parameters known as the reflection coefficients $\{K_k, 1 \leq k \leq N\}$ or partial correlation (PARCOR) coefficients [69]. Reflection coefficients result as a byproduct of solving (18) or can be computed recursively from the predictor coefficients (see, for example [86], [90]). For a stable $H(z)$, the reflection coefficients have the property

$$|K_k| < 1, \qquad 1 \leq k \leq N. \tag{20}$$

Because for values of $|K_k|$ approaching 1 the poles approach the unit circle, small changes in K_k can result in large changes in the spectrum. Therefore, for quantization purposes, the reflection coefficients are usually transformed to another set of coefficients that exhibit lower spectral sensitivity as K approaches 1. Two popular transformations are [78]

$$S_k = \frac{2}{\pi} \sin^{-1} K_k, \qquad 1 \leq k \leq N \tag{21}$$

$$G_k = \frac{1}{2} \log \frac{1+K_k}{1-K_k} = \tanh^{-1} K_k, \qquad 1 \leq k \leq N. \tag{22}$$

The parameters G_k are known as log-area-ratios (LARs) from the acoustic tube analogy of the vocal tract [8], [90] and possess the property that small changes in G_k are approximately proportional to corresponding changes in the log spectrum of $H(z)$ [131]. The mse d_2 as well as the minimax error d_∞ have been used to quantize S_k and G_k. The quantization properties of K_k, S_k, and G_k have been studied by several researchers [61], [72], [131].

An alternative distortion measure used in quantizing predictor coefficients was proposed by Itakura and Saito [68], [70]; it derives from maximum-likelihood principles. A modified form of the Itakura–Saito distortion between one vector of predictor coefficients $\boldsymbol{x} = [a(1)\ a(2) \cdots a(N)]^T$ and another vector of predictor coefficients $\boldsymbol{y}$ is given by (see [57] for variations on the Itakura–Saito distortion)

$$d_I(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{\Phi}_x (\boldsymbol{x} - \boldsymbol{y}) \tag{23}$$

where

$$\boldsymbol{\Phi}_x = \{\phi(i-k)/\phi(0), 0 \leq i, k \leq N-1\} \tag{24}$$

is the normalized autocorrelation matrix whose coefficients $\phi(i-k)$ were used in computing the vector of predictor coefficients $\boldsymbol{x}$ in (18). Since the autocorrelation coefficients in (24) are normalized by $\phi(0)$, one can show that the matrix $\boldsymbol{\Phi}_x$ and the vector $\boldsymbol{x}$ uniquely determine each other. It is important to note that $\boldsymbol{\Phi}_x$ in (23) is effectively a weighting matrix but, unlike in (12) where $\boldsymbol{W}$ is fixed, here $\boldsymbol{\Phi}_x$ changes value as $\boldsymbol{x}$ changes. Since $\boldsymbol{\Phi}_x \neq \boldsymbol{\Phi}_y$ for $\boldsymbol{x} \neq \boldsymbol{y}$, the Itakura–Saito distortion is not symmetric with respect to its arguments, i.e., $d_I(\boldsymbol{x}, \boldsymbol{y}) \neq d_I(\boldsymbol{y}, \boldsymbol{x})$. The distortion measure is not a distance or a metric. By contrast, the weighted mse distortion is a symmetric distance and metric.

Even though the computation of d_I in (23) implies a matrix multiply, the computation can be simplified considerably and reduced to a scalar (dot) product [68].

4) Perceptually Motivated Distortion Measures: For very small distortions, and therefore high bit rates, most reasonable distortion measures, including those mentioned above, all exhibit similar behavior, by linearity arguments. Furthermore, they would all be expected to correlate well with subjective judgements of speech quality. However, as bit rate decreases and distortion increases, simple distortion measures have not always correlated well with perceptual judgements. Since VQ is expected to be especially useful at low bit rates, it becomes more important to develop and use distortion measures that are correlated better with human auditory behavior. A number of perceptually based distortion measures, and others that correlate well with subjective judgements, have been used in speech coding (see, for example, [74, Appendix E], [11], [12], [94], [100], [101], [134]). If high speech quality at a given bit rate is the most important consideration in a coder design, then one would do well to consider the use of a distortion measure that correlates well with human perception.

C. Codebook Design

As mentioned above, to design our L-level codebook, we partition N-dimensional space into L cells $\{C_i, 1 \leq i \leq L\}$ and associate with each cell C_i a vector $\boldsymbol{y}_i$. The quantizer then assigns the code vector $\boldsymbol{y}_i$ if $\boldsymbol{x}$ is in C_i. A quantizer is said to be an optimal (minimum-distortion) quantizer if the distortion in (4) is minimized over all L-level quantizers. There are two necessary conditions for optimality. The first condition is that the optimal quantizer is realized by using a minimum-distortion or nearest neighbor selection rule

$$q(\boldsymbol{x}) = \boldsymbol{y}_i, \qquad \text{iff } d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_j), \quad j \neq i, \quad 1 \leq j \leq L. \tag{25}$$

That is, the quantizer chooses the code vector that results in the minimum distortion with respect to $\boldsymbol{x}$. (Ties are decided by some rule.) The second necessary condition for optimality is that each code vector $\boldsymbol{y}_i$ is chosen to minimize the average distortion in cell C_i. That is, $\boldsymbol{y}_i$ is that vector $\boldsymbol{y}$ which minimizes

$$D_i = \mathcal{E}[d(\boldsymbol{x}, \boldsymbol{y}) | \boldsymbol{x} \in C_i] = \int_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{x})\, d\boldsymbol{x}. \tag{26}$$

We call such a vector the *centroid* of the cell C_i, and we write

$$\boldsymbol{y}_i = \text{cent}(C_i). \tag{27}$$

Computing the centroid for a particular region will depend on the definition of the distortion measure. (The cells thus

Exhibit 1 Page 464

defined are known as nearest neighbor cells, Voronoi cells, or Dirichlet regions [48].)

In practice, we are given a set of training vectors $\{x(n), 1 \leq n \leq M\}$. A subset M_i of those vectors will be in cell C_i. The average distortion D_i is then given by

$$D_i = \frac{1}{M_i} \sum_{x \in C_i} d(x, y_i). \qquad (28)$$

For either the mse or the weighted mse criterion, one can show that D_i is minimized by

$$y_i = \frac{1}{M_i} \sum_{x \in C_i} x(n) \qquad (29)$$

or y_i is simply the sample mean of all the training vectors contained in C_i. For the Itakura–Saito distortion d_I, one can show that y_i is computed by first averaging the normalized autocorrelations corresponding to the sample vectors

$$\phi_{y_i}(k) = \frac{1}{M_i} \sum_{x \in C_i} \phi_x(k), \qquad 0 \leq k \leq N \qquad (30)$$

where $\phi_x(k)$ are normalized such that $\phi_x(0) = 1$. The vector y_i is then obtained as the solution to (18) with $\phi_{y_i}(k)$ as the autocorrelation coefficients.

One method for codebook design is an iterative clustering algorithm known in the pattern-recognition literature as the *K-means algorithm*.[5] In our problem here, $K = L$. The algorithm divides the set of training vectors $\{x(n)\}$ into L clusters[6] C_i in such a way that the two necessary conditions for optimality are satisfied. Below, m is the iteration index and $C_i(m)$ is the ith cluster at iteration m, with $y_i(m)$ its centroid. The algorithm is as follows:

K-Means Algorithm

Step 1: *Initialization:* Set $m = 0$. Choose by an adequate method a set of initial code vectors $y_i(0)$, $1 \leq i \leq L$. (See Section V-E.)

Step 2: *Classification:* Classify the set of training vectors $\{x(n), 1 \leq n \leq M\}$ into the clusters C_i by the nearest neighbor rule

$$x \in C_i(m), \qquad \text{iff } d[x, y_i(m)] \leq d[x, y_j(m)], \quad \text{all } j \neq i.$$

Step 3: *Code Vector Updating:* $m \leftarrow m + 1$. Update the code vector of every cluster by computing the centroid of the training vectors in each cluster

$$y_i(m) = \text{cent}(C_i(m)), \qquad 1 \leq i \leq L.$$

Step 4: *Termination Test:* If the decrease in the overall distortion $D(m)$ at iteration m relative to $D(m-1)$ is below a certain threshold, stop; otherwise go to Step 2.

Any other reasonable termination test may be substituted for Step 4 above.

The above algorithm can be shown to converge to a local optimum (see, for example, [5], [81]). Furthermore, any such solution is, in general, not unique [33], [58]. Global optimality may be achieved approximately by initializing the code vectors to different values and repeating the above algorithm for several sets of initializations and then choosing the codebook that results in the minimum overall distortion. In Section V-E we shall discuss methods for performing initialization.

D. *Computational and Storage Costs*

Having designed a codebook as described above, one can then use it to quantize each input vector $x(n)$. The quantization is performed as shown in (25) by computing the distortion between $x(n)$ and each of the code vectors, then choosing the code vector with the minimum distortion as the quantized value of $x(n)$. This type of quantization is known as a *full search* since all code vectors are tested for quantizing each input vector. For an L-level quantizer, the number of distortion computations needed to quantize a single input vector is L. While a distortion computation can be arbitrarily complex, we shall assume here that each distortion computation requires a total of N multiply-adds (this is true for the mse and one version of the Itakura–Saito distortion). Therefore, the *computational cost* for quantizing each input vector is

$$\mathscr{C} = NL. \qquad (31)$$

If we encode each code vector into $B = RN = \log_2 L$ bits for transmission, then

$$\mathscr{C} = N\,2^{RN}. \qquad (32)$$

Thus computation cost grows exponentially with the number of dimensions and the number of bits per dimension.

Another important part of the quantization cost is *memory* or *storage cost*, i.e., how much storage is needed to store the code vectors. We shall measure storage assuming one storage location per vector parameter. It is clear that storage cost is then given by

$$\mathscr{M} = NL = N\,2^{RN}. \qquad (33)$$

Like computational cost, storage cost is exponential in the number of dimensions and the number of bits per dimension.

The costs given above are for the full search algorithm. In Section V we shall present methods that reduce computational costs substantially at the cost of relatively small loss in performance and/or increase in storage.

While we place heavy emphasis on the costs associated with the quantization process itself, it is also important to

[5] The algorithm presented here was described by Forgy in 1965 [41] and is the clustering algorithm described most in the pattern-recognition literature [5], [30], [64], [99], [129]. The name *K-means* comes from MacQueen [84], who actually describes a different algorithm. In an unpublished paper in 1957, Lloyd had independently developed the same algorithm as Forgy's but for the scalar quantization problem and a known distribution (Lloyd's paper has recently been published [82].) The application of this algorithm to a training sequence and the VQ case has been termed in some of the information theory literature as the generalized Lloyd algorithm [56]. Linde, Buzo, and Gray [81] have shown that the algorithm works with a large class of distortion measures, including measures that are not metric, and so the algorithm has also been called the LBG algorithm.

[6] We use the same symbol C_i to represent both the cluster and the cell corresponding to code vector y_i. Cell C_i is that region of N-dimensional space that is closest to y_i based on the nearest neighbor rule, while cluster C_i is that subset of training vectors which are closest to y_i based on the same rule, i.e., cluster C_i is the set of training vectors contained in cell C_i.

keep in mind the costs associated with design of the codebook in the first place. In the K-means algorithm described above, most of the computations result from the classification step; code vector updating presents a negligible amount of computation by comparison. For an L-level quantizer, M training vectors, and I iterations, the computational cost for training is

$$\mathscr{C}_T = NLMI = N2^{NR}MI. \qquad (34)$$

The storage cost, including the storage needed to store all the training vectors, is

$$\mathscr{M}_T = N(L + M). \qquad (35)$$

For reliable design of the codebook, it has been our experience that one needs at least 10 and preferably about 50 training vectors per code vector, so that M is on the order of $10L$ or more (see Section V-E). So, storage cost for training is largely dominated by the amount of training vectors needed.

E. Vector Quantization Model

Before we delve into more mathematics and examine the detailed workings of vector quantization, we shall present a simple model of vector quantization, which we hope will give the reader a basic understanding of the various processes at work. The concepts presented will be illustrated by simplified examples.

1) Basic Model: Our VQ model identifies four properties of vector components which, when utilized appropriately in codebook design, result in optimal performance. The four properties of vector components are: *linear dependency*, *nonlinear dependency*, *pdf shape*, and *dimensionality*. These four properties are interrelated to a certain extent. For example, even though the multidimensional pdf shape completely specifies any dependencies among vector components, we shall see that pdf shape still plays an important role in determining optimal performance, even when all dependencies among components are removed. For a given codebook size, VQ takes advantage of the four properties above by proper *placement* of the code vectors in N-dimensional space. By code vector placement we mean having the freedom to place code vectors where they are needed most so as to minimize the given distortion measure. For example, one would not place code vectors in regions of zero probability. There are two aspects of code vector placement that are of interest: code vector *spacing* or density and *cell shape*. Code vector spacing refers to how close the code vectors are to each other. In general, one would expect closer spacing (higher density) in high pdf regions (i.e., where $p(\mathbf{x})$ is large) and wider spacing of code vectors in regions of low pdf. Once the code vectors are specified, then the cell shapes are automatically determined by the distortion measure. Conversely, once all the cell shapes and positions are specified, then the code vectors are automatically determined as the cell centroids. We shall see below that it will be beneficial to think in terms of cell shapes to gain a better understanding of the workings of VQ, for it is the freedom we have to choose different cell shapes in higher dimensions ($N > 1$) that allows us to exploit dimensionality to minimize distortion in a way that is not possible with scalar quantization. Below, we demonstrate how a vector quantizer chooses its code vector placements and cell shapes, taking advantage of the four vector properties to optimize performance. The examples are designed to expose the processes at work in VQ in a simple way and to show how they may differ from scalar quantization.

2) Dependency: Data compression is largely a process of redundancy removal; it is not necessary to waste bits in transmitting redundant information. Redundancy usually implies some sort of dependency among transmission parameters. We shall classify statistical dependency into two types: *linear dependency* and *nonlinear dependency*. These terms are explained below.

Linear dependency is what we normally think of as *correlation*. Two random variables that are correlated are linearly dependent. If the variables are uncorrelated, they are no longer linearly dependent, but they may still be statistically dependent. The latter "residual" dependency we call nonlinear dependency; it is whatever dependency remains after the linear dependency is removed. Two (zero-mean) variables x_1 and x_2 are uncorrelated if the expected value of their product is zero

$$\mathscr{E}[x_1x_2] = 0 \quad \text{(uncorrelated)}. \qquad (36)$$

But x_1 and x_2 are independent if and only if their joint pdf is equal to the product of the individual (marginal) densities of x_1 and x_2

$$p(x_1, x_2) = p(x_1)p(x_2), \qquad \text{for all } x_1, x_2 \quad \text{(independent)}. \qquad (37)$$

If x_1 and x_2 are uncorrelated but dependent (i.e., (36) applies but not (37)), then that dependency we call nonlinear. We shall now demonstrate how one can take advantage of both types of dependency in reducing the necessary bit rate for transmission. In the examples below we shall use the mse as our distortion measure.

EXAMPLE 1

x_1 and x_2 are two random variables whose joint pdf $p(x_1,x_2)$ is shown in Fig. 5. The density is uniform and nonzero inside the rectangle and zero outside the rectangle

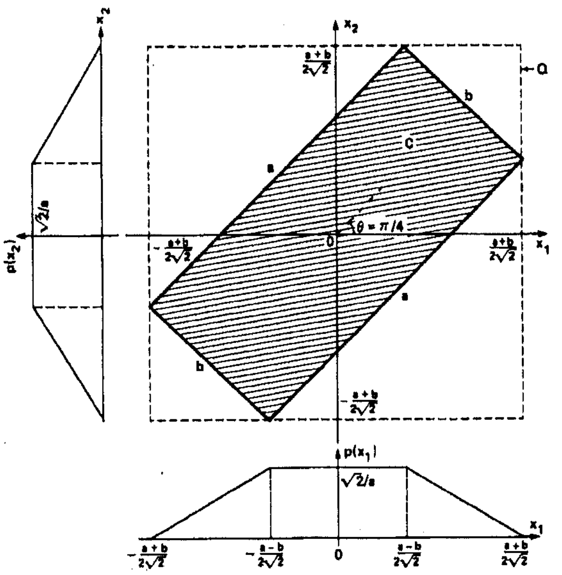

Fig. 5. An example of a uniform pdf $p(x_1,x_2)$ in two dimensions; the density is zero outside the region C. Shown also are the marginal densities $p(x_1)$ and $p(x_2)$. x_1 and x_2 in this example are correlated.

gle. Let the region inside the rectangle be denoted by C, then

$$p(x_1, x_2) = p(\mathbf{x}) = \begin{cases} \frac{1}{ab}, & \mathbf{x} \in C \\ 0, & \text{otherwise.} \end{cases} \quad (38)$$

The rectangle has sides a and b, and is oriented at an angle $\theta = \pi/4$. Shown also in Fig. 5 are the marginal densities $p(x_1)$ and $p(x_2)$, which in this example happen to be equal. It is clear from (38) and the marginal densities in Fig. 5 that (37) does not apply, and so x_1 and x_2 are dependent. One can also show that x_1 and x_2 are correlated, so that (36) is not true either.

Now, let us use a scalar *uniform quantizer* to quantize x_1 and x_2 independently. A uniform quantizer is one whose quantizing intervals C_i are of equal length. (Such a quantizer minimizes the mse only for a uniform density and, therefore, would not be optimal for quantizing x_1 and x_2 in this example.) We shall quantize x_1 and x_2 using quantizing intervals equal to Δ. Since x_1 and x_2 have values that range between $-(a+b)/2\sqrt{2}$ and $(a+b)/2\sqrt{2}$, the number of levels needed to quantize each of x_1 and x_2 is equal to

$$L_1 = L_2 = \frac{a+b}{\sqrt{2}\Delta}. \quad (39)$$

x_1 and x_2 can be coded using $R_1 = \log_2 L_1$ bits and $R_2 = \log_2 L_2$ bits, respectively. The vector $\mathbf{x}$ can be coded, then, using

$$B_x = R_1 + R_2 = \log_2 L_1 L_2 = \log_2 \frac{(a+b)^2}{2\Delta^2} \text{ bits.} \quad (40)$$

The two scalar quantizers correspond to using a vector quantizer with a total number of levels

$$L_x = L_1 L_2 = \frac{(a+b)^2}{2\Delta^2}. \quad (41)$$

Indeed, such a quantizer can be obtained by first demarking the extreme values of x_1 and x_2 by the dashed rectangle (it is a square in this example) enclosing the region Q in Fig. 5, and then drawing a rectangular grid with the separation between grid lines equal to Δ in both directions (see Fig. 8). Such a quantizer would have quantization cells in the form of squares, each of area Δ^2. Clearly, the total number of such squares inside the dotted region Q is obtained by dividing the total area by Δ^2. The result is equal to L_x in (41). Such a vector quantization code is known as a *product code* because it is the Cartesian product of the codes for quantizing x_1 and x_2 separately. The use of a product code for this example is clearly wasteful of bits since regions of zero probability are assigned some of the bits.

We have seen above that, with a product code, the total number of quantization levels is proportional to the area of the whole quantization region (region Q in Fig. 5). Therefore, if the area of the quantization region can somehow be reduced, the number of quantization levels and the corresponding bit rate will also be reduced. For our example, one can perform a coordinate transformation via a rotation that transforms Fig. 5 into Fig. 6. The vector $\mathbf{x}$ is transformed into another vector $\mathbf{u}$. One can show that the new coordinates u_1 and u_2 are, in fact, uncorrelated. With the proper rotation, any set of random variables can be rendered uncorrelated, as we shall see in Section IV. In the special case of the example in Fig. 6, we see from the marginal densities shown in the figure that

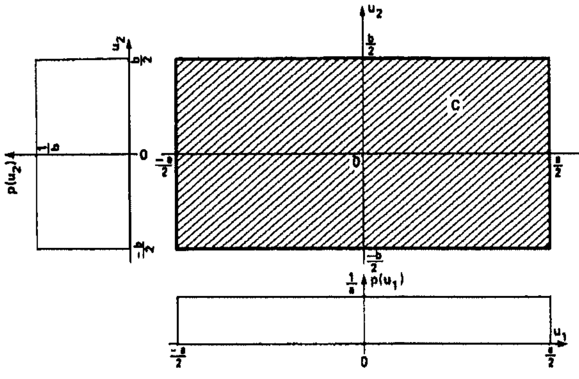

Fig. 6. The pdf in Fig. 5 after a rotation of coordinates. u_1 and u_2 are now uncorrelated and independent.

$$p(u_1, u_2) = p(u_1)p(u_2), \qquad \text{for all } u_1, u_2$$

and, therefore, u_1 and u_2 are also statistically independent. Performing uniform scalar quantization in this case with a quantizing interval of length Δ yields a number of levels equal to

$$L_1 = \frac{a}{\Delta} \qquad L_2 = \frac{b}{\Delta}$$
$$L_u = L_1 L_2 = \frac{ab}{\Delta^2}. \quad (42)$$

The corresponding number of bits is

$$B_u = \log_2 \frac{ab}{\Delta^2}. \quad (43)$$

The difference in the number of bits needed to code $\mathbf{x}$ and $\mathbf{u}$ can be seen from (40) and (43) to be

$$B_x - B_u = \log_2 \frac{(a+b)^2}{2ab}. \quad (44)$$

For example, for $a = 2b$

$$B_x - B_u = 1.17 \text{ bits} \qquad (a = 2b). \quad (45)$$

Therefore, the rotation saves us in excess of 1 bit per transmitted vector. Such a difference can become significant at low data rates.

In comparing bit rates in (44) we made an implicit assumption that the total quantization distortion is the same for both examples of Figs. 5 and 6. Otherwise, comparing bit rates as such is not meaningful. In fact, for small Δ and, hence, large B_x and large B_u, one can show from (4) that the total distortion in both cases is about the same. Differences in distortion arise as Δ increases and boundary effects near the edges of the rectangle become significant. Under such circumstances one must be careful to equalize the distortions before comparing bit rates. At such low bit rates, (45) would not be expected to hold; it would most likely decrease.

EXAMPLE 2

Example 1 demonstrated how one can take advantage of decorrelation through rotation to reduce the bit rate in scalar quantization of a vector. In this example, we shall demonstrate how, through vector quantization, one can take advantage of nonlinear dependencies to reduce the bit rate.

Exhibit 4 Page 467

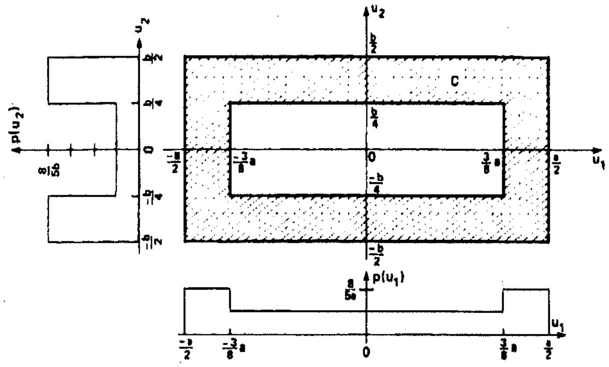

Fig. 7. An example where u_1 and u_2 are uncorrelated but dependent; this type of dependency is termed *nonlinear.*

Fig. 7 shows a different example of a pdf that is nonzero and uniform inside the hatched region C and zero otherwise. Since the area of the hatched region is $5ab/8$, we have

$$p(u) = \begin{cases} \dfrac{8}{5ab}, & u \in C \\ 0, & \text{otherwise}. \end{cases} \tag{46}$$

One can show that, like the example in Fig. 6, u_1 and u_2 are uncorrelated but, unlike in Fig. 6, u_1 and u_2 here are not independent, as is clear from (46) and the marginal densities in Fig. 7. Such statistical dependency is termed nonlinear; it cannot be removed by a process of decorrelation. A scalar quantizer designed for u_1 and u_2 in Fig. 7 will yield the same bit rate B_u in (43). To take advantage of the nonlinear dependency we must use a different vector quantizer that partitions only the hatched area in Fig. 7 and does not waste bits inside the small rectangle. One such quantizer would divide only the hatched area into squares of equal area Δ^2. (Such a quantizer would yield the same distortion as the scalar quantizer for small Δ.) The number of levels and bits for this vector quantizer would be

$$L'_u = \frac{5}{8}\frac{ab}{\Delta^2} \qquad B'_u = \log_2 \frac{5ab}{8\Delta^2}. \tag{47}$$

The reduction in bit rate between a scalar quantizer and a vector quantizer in this case is the difference between (43) and (47)

$$B_u - B'_u = \log_2 \frac{8}{5} = 0.68 \text{ bits}. \tag{48}$$

Therefore, taking advantage of nonlinear dependency in this case saved us 0.68 bits/vector.

We have seen how, with proper cell or code vector placement, a vector quantizer was able to take advantage of nonlinear dependencies to reduce the bit rate. It should be clear that any vector rotation will not affect the vector quantizer's ability to locate its cells properly. Therefore, VQ can make effective use of all dependencies (linear or nonlinear) simply by proper code vector placement.

3) Dimensionality:

EXAMPLE 3

The vector quantizer in Example 2 employed the square as the shape of all its cells. The square shape was inspired from the scalar quantizers used earlier. But one property of vector quantizers in higher dimensions is that one has the freedom to choose other cell shapes and not be restricted to the N-dimensional cubes suggested by scalar quantization. Let us examine the effect of using a different shape, such as the *hexagon,* to partition our two-dimensional space. Fig. 8 shows a covering of space by squares and Fig. 9 shows a space covering by regular hexagons. Let each

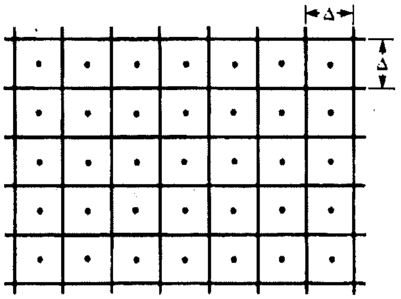

Fig. 8. Packing of two-dimensional space with squares.

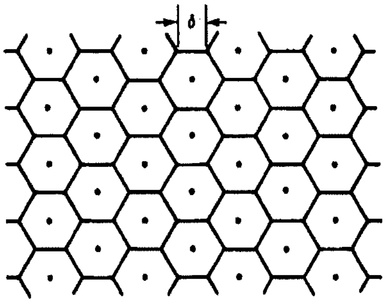

Fig. 9. Packing of two-dimensional space with regu hexagons. For the same number of cells in a given area w a uniform pdf, hexagons have a lower mse than squares.

hexagon have sides of size δ. The area of the hexagon is then

$$A_H = \frac{3\sqrt{3}}{2}\delta^2. \tag{49}$$

To compare the performance of the square quantizer to the hexagon quantizer, we must compare the quantization mse for both quantizers. If we assume that the code vectors are located in the center of the square and the hexagon in each case (as shown by the dots in Figs. 8 and 9), one can show that the mse for each cell is given by

$$E_S = \frac{\Delta^4}{6} \qquad \text{(square)} \tag{50}$$

$$E_H = \frac{5\sqrt{3}}{8}\delta^4 \qquad \text{(hexagon)} \tag{51}$$

where the subscripts denote the respective quantizers. The total mse is then obtained by multiplying (50) and (51) by the number of cells (levels). If we make the area of the hexagon equal to the area of the square, i.e., set $A_S = \Delta^2 = A_H$ in (49), then, neglecting edge effects, both quantizers will have the same number of cells covering any given area and, therefore, they will have the same bit rate. However, the ratio of the distortions can be shown to be

$$\frac{E_H}{E_S} = \frac{5\sqrt{3}}{9} = 0.962 \qquad (A_S = A_H) \tag{52}$$

Exhibit 4 Page 468

which is equivalent to −0.167 dB. That is, the hexagon quantizer will yield a lower mse than the square quantizer by a fraction of a decibel. Conversely, if we equalize the distortions, i.e., set $E_S = E_H$, then the ratio of the areas will be

$$\frac{A_H}{A_S} = \frac{3}{\sqrt[4]{3}\sqrt{5}} = 1.0194 \qquad (E_S = E_H). \tag{53}$$

This means that, for the same distortion, the hexagon has a slightly larger area than the square and, therefore, will require proportionately fewer cells to cover the same space. Therefore, the bit rate for the hexagon will be lower by an amount equal to $\log_2 A_H/A_S$, which from (53) is equal to 0.028 bits.

The gain of 0.028 bits resulting from the use of a hexagon quantizer can be obtained in the examples of Figs. 6 and 7 in addition to any other gains obtained by taking advantage of linear and nonlinear dependencies. This means that, even in the case when two random variables are independent, as in Fig. 6, vector quantization can squeeze out an additional small fraction of a bit by taking advantage of the higher dimensionality using an appropriate cell shape.

4) PDF Shape: In the examples above, the cells always had not only the same shape throughout the quantization region but also the same size. Effectively, the cell *spacing* was uniform. Uniform cell spacing is certainly a reasonable choice with a uniform pdf. However, for nonuniform pdf shapes, one would expect that some form of nonuniform cell spacing would be needed for optimal performance. We shall see in Section III how scalar and vector quantizers make effective use of pdf shape to reduce the bit rate.

In this section, we presented a model for vector quantization and gave a few examples to illustrate some of the basic principles. In the following sections, we shall review some of the theoretical foundations of vector quantization and give practical examples from speech coding. However, the basic concepts presented in the vector quantization model above and in the simplified examples will still hold when we deal with real-world data.

III. Theoretical Vector Quantizer Performance

In this section we review some of the important *theoretical* tools that are used to help estimate the performance of vector quantizers, along with some of the known results. The presentation is under three headings: rate-distortion theory, scalar quantization, and asymptotic vector quantization. Rate-distortion theory and scalar quantization will give us lower and upper bounds, respectively, on the minimum bit rate achievable by vector quantizers for any given distortion.

This section is theoretical in nature and is rather long. At first reading, we recommend that the reader take only a cursory look at the contents of this section to gain familiarity with the basic definitions and concepts presented, and move quickly to Section IV and the remainder of the paper.

A. Rate-Distortion Theory

Since a major purpose in performing data compression is to minimize the bit rate for a desired level of distortion, it is important in a particular situation to know the theoretical lower bound on the bit rate for any quantizer. By knowing such a bound, one can compare the performance of different quantizers to that bound and decide whether to search for other possibly more complex quantizers that might approach that bound more closely. Rate-distortion theory, a branch of information theory, deals with obtaining such lower bounds without requiring the design of actual quantizers. For a given distortion D, one can compute either $R(D)$, the *rate-distortion function*, defined as the minimum achievable rate (per dimension) for a given distortion D, or its inverse $D(R)$, the *distortion-rate function*, defined as the minimum achievable distortion for a given rate R. The performance limit provided by $D(R)$ or $R(D)$ applies to all methods of source coding, not just vector quantization. It specifically allows for coders that incorporate arbitrarily long delays. In light of the complexity of encoding permitted by this theory, not being able to come very close to the optimal performance when investigating practical vector quantizers may not be indicative of a meaningful disparity. It is when the performance of practical coders is relatively close to the rate-distortion limits that the theory is most useful. Below, we present a summary of the relevant results from rate-distortion theory, preceded by an introduction to the concept of *entropy* from information theory.

1) Coding of Quantizer Output: We mentioned in Section II that one can code the quantizer output vectors $\{y_i, 1 \leq i \leq L\}$ with $\log_2 L$ bits each. If L is a power of 2, the coding is very simple, but if L is not a power of 2, then one can group several vectors together and then perform the coding. For example, if $L = 5$, then to code a single vector would require 3 bits instead of the desired $\log_2 5 = 2.32$ bits, since one cannot transmit a fractional number of bits as such. However, if we group three vectors together, the total number of levels is equal to $L^3 = 125 \cong 2^7$, then each triplet can be coded using 7 bits, for an average of 2.33 bits which is close to the desired average. Henceforth, we shall disregard the problem of fractional bits and simply assume that values can be grouped together, if desired, to achieve the needed rate.

In general, coding L vectors with $\log_2 L$ bits each is actually the maximum rate needed for coding. The minimum average achievable rate to code the vectors $\{y_i\}$ is given by the *entropy* of $\{y_i\}$ [46], defined as

$$H(y) = -\sum_{i=1}^{L} P(y_i) \log_2 P(y_i) \tag{54}$$

where $H(y)$ is the entropy of the discrete-amplitude variable y and $P(y_i)$ is the discrete probability of y_i. $H(y)$ is also called *self-information*, and it is a measure in bits of the information in $\{y_i\}$. Since all discrete probabilities must obey

$$0 \leq P(y_i) \leq 1 \qquad \sum_{i=1}^{L} P(y_i) = 1 \tag{55}$$

one can show [46] that entropy is bounded by

$$0 \leq H(y) \leq \log_2 L. \tag{56}$$

Each vector y_i is coded using

$$B_i = -\log_2 P(y_i) \quad \text{bits} \tag{57}$$

so that vectors with different probabilities will have different wordlengths. The resulting code will be a *variable-length* code, with an average rate equal to the entropy $H(y)$ in (54). This type of coding is known as *entropy coding*. The Huffman code [66] is a well-known, straightforward method

Exhibit 4 Page 469

for performing entropy coding. With variable-length coding, appropriate buffering schemes would need to be implemented if transmission is over a fixed-rate channel. Any such buffering would, of course, introduce a delay in the system. Also, variable-length coding is particularly sensitive to channel errors. *Permutation codes* [15] offer an alternative for entropy coding using fixed-length codes; however, the codewords can be very long, which also results in delays. In practice, variable-length codes with buffering are used.

One can show that maximum entropy is achieved when all vectors y_i have equal probability [46], i.e., $P(y_i) = 1/L$, in which case

$$H_{\max}(y) = B_{\max} = \log_2 L \quad \text{bits.} \tag{58}$$

For this special case, a fixed-length code is, in fact, optimal.

Entropy coding is one form of *noiseless coding* in that the coding does not introduce any additional noise or distortion beyond that introduced by the quantization process; it merely takes advantage of the probability distribution to minimize the bit rate. (The purpose of the noiseless encoding box in Fig. 1 is to minimize the bit rate without introducing extra distortion.)

One important and useful property of entropy coding is that, even if the number of levels L is infinite, the entropy can still be finite. (Values with very small probability contribute very little to the entropy since the limit as $P \to 0$ of $-P\log_2 P$ is zero.) Therefore, one can partition N-dimensional space into a countably infinite number of finite (nonzero) cells, and the entropy of the resultant codebook will be finite. In practice, L must be finite, but it could be made large without substantially increasing the bit rate, provided entropy coding is used.

As the cells in the quantization process are made smaller so that their size goes to zero, the quantizer discrete outputs $\{y_i\}$ tend to the original continuous-amplitude source x; the quantization distortion goes to zero; and the entropy becomes infinite. In other words, it takes infinitely many bits to transmit a continuous-amplitude source. For such a source, it will prove useful to define its *differential entropy* [14]

$$h(x) = -\int_x p(x)\log_2 p(x)\,dx \tag{59}$$

where $h(x)$ is the differential entropy of the vector x and the integral is over all the components of x. Unlike *absolute* entropy, which was defined in (54), differential entropy may be positive or negative. As such, differential entropy values are not meaningful in and of themselves; they are meaningful only relative to other differential entropies. Thus the difference between two differential entropies is a meaningful measure of the difference in information (in bits) between the corresponding sources.

For a random variable x with a given variance σ^2, one can show that $h(x)$ is bounded above by [122]

$$h(x) \leq h_G(x) = \tfrac{1}{2}\log_2(2\pi e\sigma^2) \tag{60}$$

where $h_G(x)$ is the differential entropy of a Gaussian pdf with variance σ^2. We shall see that Gaussian sources play a special role in bounding the performance of coding systems.

2) Rate-Distortion Theory Results: Most of the results of rate-distortion theory have been obtained for a scalar source that is defined as a function of a discrete variable (such as sampled time). Let $x(n)$ be a discrete-time, continuous-amplitude random source, i.e., a stochastic sequence. If all samples of $x(n)$ are independent and identically distributed, we say that the source is *memoryless* and is completely specified by a single pdf, $p(x)$. It would appear that in quantizing a memoryless source, vector quantization should not have any advantages over scalar quantization. However, we have already seen in Example 3 that VQ indeed can achieve better performance even in the case of a memoryless source. Therefore, in investigating rate-distortion limits we group samples of the source in blocks or vectors and determine the conditions that achieve those limits.

We shall group N consecutive values of $x(n)$ into a single vector x, so that x is a random vector. The vector x is then quantized into a vector $y = q(x)$, where y is one of the vectors in the set $\{y_i, 1 \leq i \leq L\}$, with L possibly infinite. The average distortion D in representing x as y is given by $\mathcal{E}[d(x,y)]$, where $d(x,y)$ is the distortion per dimension (i.e., per sample of $x(n)$). The vectors y can be transmitted at an average bit rate of $R = H(y)/N$ bits per sample, where $H(y)$ is the entropy of y as defined in (54). The minimum achievable distortion $D_N(R)$ for a given rate R is given by

$$D_N(R) = \min_{q(x)} \mathcal{E}[d(x,y)], \qquad \text{with } \frac{1}{N}H(y) \leq R \tag{61}$$

where the minimum is taken over all possible mappings $q(x)$. The distortion-rate function $D(R)$ is obtained in the limit as $N \to \infty$

$$D(R) = \lim_{N\to\infty} D_N(R). \tag{62}$$

$D(R)$ is the minimum attainable distortion in coding the source $x(n)$ at a rate R; it is a lower bound on the performance of any quantization scheme. The rate-distortion function $R(D)$ is the inverse of $D(R)$ and is defined similarly.[7] We shall have occasion to use both functions in this paper, although $D(R)$ is used more in practice since we are typically given the rate R and we design our quantizer to minimize the distortion.

The main result of rate-distortion theory that relates to VQ is that, *by using a vector quantizer, one can in principle approach the distortion-rate function arbitrarily closely by increasing the vector size N.* This is essentially implied in the definition of $D(R)$ in (62). Therefore, $D(R)$ is not merely a lower bound for any quantizer, it is actually achievable, in theory, by a vector quantizer of high dimension.

The distortion-rate function $D(R)$ has two important properties: it is *monotone decreasing* with R and it is *convex*. Furthermore, for the mse distortion, $D(R)$ decreases at the rate of about 6 dB/bit for large R. We shall exhibit these properties in specific examples below.

While defining $R(D)$ or $D(R)$ is straightforward, neither is simple to evaluate analytically, except for a few special cases. For a memoryless (zero-mean) Gaussian source with variance σ^2 and a mse distortion, $R(D)$ and $D(R)$ are

[7]The general definition of $R(D)$ in rate-distortion theory is given in terms of the concept of *mutual information* [14]. We have chosen here a somewhat narrower definition that is sufficient for our purposes.

Exhibit 1 Page 470

given by [14]

$$R_G(D) = \max\left\{0, \tfrac{1}{2}\log_2 \frac{\sigma^2}{D}\right\}$$

$$= \begin{cases} \tfrac{1}{2}\log_2 \sigma^2/D, & 0 \leq D \leq \sigma^2 \\ 0, & D > \sigma^2 \end{cases} \quad (63)$$

$$D_G(R) = 2^{-2R}\sigma^2. \quad (64)$$

If $D > \sigma^2$ is given, then we need not transmit any information ($R = 0$) because we can always obtain $D = \sigma^2$ by using zeros for the quantized values of $x(n)$, since the quantization error is then equal to $x(n)$. By dividing (64) by σ^2 we obtain the *normalized distortion* which can be measured in decibels

$$\mathscr{D}_G(R) = 10\log_{10}\frac{D_G(R)}{\sigma^2} = -6.02R \text{ dB} \quad (65)$$

where *script* $\mathscr{D}$ is the normalized distortion in decibels. The negative of $\mathscr{D}$ is just the signal-to-noise ratio (SNR) in decibels of the quantizer

$$\text{SNR} = -\mathscr{D} = 10\log_{10}\frac{\sigma^2}{D} \text{ dB}. \quad (66)$$

For much of this paper we shall use $\mathscr{D}$ instead of SNR to measure quantizer mse distortion.

There are scant explicit $D(R)$ results for memoryless non-Gaussian sources (see the result for Laplacian densities by Berger [14]). However, lower and upper bounds exist

$$D^*(R) \leq D(R) \leq D_G(R). \quad (67)$$

Thus $D(R)$ is upper bounded by the distortion-rate function for a Gaussian source. The lower bound $D^*(R)$, known as the *Shannon lower bound*, for a mse distortion is given by

$$D^*(R) = \frac{1}{2\pi e}2^{2h(x)}2^{-2R} \quad (68)$$

$$R^*(D) = h(x) - \tfrac{1}{2}\log_2 2\pi e D \quad (69)$$

where $h(x)$ is the differential entropy of the memoryless source. The Shannon lower bound is achievable for many sources only as $R \to \infty$. From (60) and (68), we can write $\mathscr{D}^*(R)$ as a normalized distortion in decibels

$$\mathscr{D}^*(R) = -6.02R - 6.02[h_G(x) - h(x)] \text{ dB} \quad (70)$$

or

$$\mathscr{D}_G(R) - \mathscr{D}^*(R) = 6.02[h_G(x) - h(x)] \text{ dB}$$
$$= 6.02[R_G(D) - R^*(D)] \text{ dB}. \quad (71)$$

Since $h_G(x) > h(x)$, $\mathscr{D}^*(R)$ is less than the Gaussian distortion-rate function by an amount equal to the difference between the source and Gaussian differential entropies (in bits) multiplied by 6.02 dB/bit. Equation (70) makes it very clear that the asymptotic behavior of the distortion-rate function for many sources is expected to decrease at a rate of -6.02 dB/bit as $R \to \infty$.

Table 1 shows four pdfs that are common models used for certain signal distributions. Shown are the differential entropies, the difference $R_G(D) - R^*(D)$, and the corresponding difference in distortion. The Gamma pdf shows the greatest deviation from the Gaussian; it is by far the sharpest or most peaked of the four pdfs, and it becomes unbounded at $x = 0$. The Gamma density is often mentioned as a good model for the first-order pdf of speech [74, p. 32]. However, this model is only good for long-term statistics. Medium-term statistics (on the order of 100 ms), where the speech amplitudes are normalized with respect to the medium-term energy, show that the Laplacian pdf becomes a better model of speech [96]. Short-term statistics (on the order of 20 ms) show the Gaussian pdf to be a good first-order model [96]. The Gaussian pdf also appears to be a good short-term first-order model for the prediction residual in adaptive predictive coding [7], [74]. Since any speech coding system operating at 2 bits/sample or less would need to be normalized with respect to short-term energy to minimize distortion, we shall assume in this paper that the first-order pdf for speech and the prediction residual is essentially Gaussian.

Table 1 Four Common PDFs and their Differential Entropies $h(x)$. Column 4 gives the difference in bits between the Gaussian rate-distortion function $R_G(D)$ and the Shannon lower bound $R^*(D)$ for each pdf. Column 5 shows the corresponding difference in distortion in decibels; it is obtained by multiplying column 4 by 6.02. Column 6 is the asymptotic (high bit rate) difference in bits between the rate of a Lloyd–Max quantizer R_{LM} and the Shannon lower bound, and column 7 is the corresponding difference for the distortion. (From Jayant and Noll [74].)

pdf	$p(x)$	$h(x)$	$R_G(D) - R^*(D) = h_G(x) - h(x)$ (bits)	$\mathscr{D}_G(R) - \mathscr{D}^*(R)$ (dB)	$R_{LM} - R^*$ (bits)	$\mathscr{D}_{LM} - \mathscr{D}^*$ (dB)
Gaussian	$\frac{1}{\sqrt{2\pi}\,\sigma}\exp[-x^2/2\sigma^2]$	$\frac{1}{2}\log_2(2\pi e\sigma^2)$	0	0	0.722	4.35
Uniform	$\frac{1}{2\sqrt{3}\,\sigma}$, $\lvert x\rvert \leq \sqrt{3}\,\sigma$; 0, otherwise	$\frac{1}{2}\log_2(12\sigma^2)$	0.255	1.53	0.255	1.53
Laplacian	$\frac{1}{\sqrt{2}\,\sigma}\exp[-\sqrt{2}\lvert x\rvert/\sigma]$	$\frac{1}{2}\log_2(2e^2\sigma^2)$	0.104	0.63	1.190	7.17
Gamma	$\frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma\lvert x\rvert}}\exp[-\sqrt{3}\lvert x\rvert/2\sigma]$	$\frac{1}{2}\log_2(4\pi e^{1-C}\sigma^2/3)$; C = Euler's constant = 0.5772	0.709	4.27	1.963	11.82

Exhibit 4 Page 476

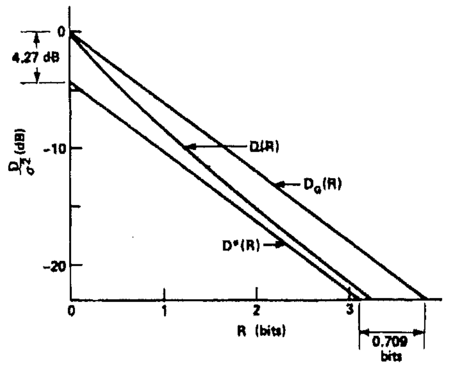

Fig. 10. Distortion-rate function $D(R)$ for a memoryless Gamma source and a mse distortion measure, bounded above by the Gaussian distortion-rate function $D_G(R)$ and below by the Shannon lower bound $D^*(R)$, from Noll and Zelinski [97]. $D_G(R)$ and $D^*(R)$ are always parallel straight lines in the decibel scale with a slope of -6.02 dB/bit. $D(R)$ tends toward a slope of -6.02 dB/bit at high bit rates (usually $R > 3$ bits).

Fig. 10 shows a plot of $D(R)$ for the Gamma pdf along with the Gaussian upper bound $D_G(R)$ and the Shannon lower bound $D^*(R)$. (The reason for choosing the Gamma pdf is because it shows very clearly the departure from the Gaussian.) The $D(R)$ plot was obtained using Blahut's algorithm [17]. Note that the $D(R)$ curve is monotone decreasing and convex. Also, as R increases beyond a few bits, $D(R)$ decreases at the rate of about 6.02 dB/bit which is the slope of the upper and lower bounds. The slope of -6.02 dB/bit will recur over and over again for many types of quantizers as R increases.

Thus far we have considered only memoryless sources. For sources with memory, where the samples are dependent, explicit $D(R)$ results are even more meager. What little is known is for the Gaussian case where the dependence is linear and can be specified completely by the spectral density $\Phi(\omega)$ of the stochastic sequence $x(n)$. For this special case, $D(R)$ is given parametrically by [14]

$$D_G(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min\{\theta, \Phi(\omega)\}\, d\omega \tag{72}$$

$$R_G(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \max\left\{0, \frac{1}{2}\log_2\frac{\Phi(\omega)}{\theta}\right\} d\omega. \tag{73}$$

For the case of small distortions defined by

$$\theta \leq \min_{\omega}\{\Phi(\omega)\} \tag{74}$$

$D_G(R)$ is given by

$$D_G(R) = \gamma 2^{-2R}\sigma^2 \tag{75}$$

where

$$\gamma = \frac{1}{\sigma^2}\exp\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}\log\Phi(\omega)\, d\omega\right]$$
$$= \frac{\mathrm{GM}\{\Phi(\omega)\}}{\mathrm{AM}\{\Phi(\omega)\}} \leq 1. \tag{76}$$

GM and AM are the geometric mean and arithmetic mean, respectively. (σ^2 = arithmetic mean of the spectrum.) γ is a *spectral flatness measure* which is a nonnegative quantity that is equal to one if and only if the spectrum is flat [86], [90]. For a flat spectrum, i.e., a white source, the Gaussian signal values are independent and $D_G(R)$ in (75) becomes equal to (64). Therefore,

$$D_G(R)|_{\text{correlated source}} = \gamma D_G(R)|_{\text{white source}} \tag{77}$$

for distortions obeying (74). Note that for this case of small distortions, the distortion in (72) is equal to θ for all frequencies, which means that the reconstruction error will have a flat spectral density: a result that we shall meet again below.

Equation (77) states that $D(R)$ for a correlated Gaussian source is less than that for the corresponding white (memoryless) source. The difference for small distortions (or high rates) is equal to $-10\log_{10}\gamma$ decibels. Equation (77) quantifies the intuitive notion that one can transmit dependent sources at a lower distortion than independent sources. This general notion applies to non-Gaussian sources as well. The $D(R)$ function for such sources is still bounded above and below as in (67), where the Shannon lower bound is still defined by (68). However, $h(x)$ is now the differential entropy for the dependent source, defined as

$$h(x) = \lim_{N\to\infty}\frac{1}{N}h(\mathbf{x})$$

which will be smaller than the differential entropy for the corresponding memoryless source.

B. *Scalar Quantization*

We showed above how the distortion-rate function $D(R)$ provides a *lower bound* on the minimum distortion achievable by a vector quantizer at a given bit rate. Scalar quantizers provide us with an *upper bound* on the minimum achievable distortion. The difference between the two bounds gives the reduction in distortion that is potentially attainable by vector quantization. In this section we shall consider the scalar quantization of memoryless sources. The scalar quantization of correlated sources is discussed in Section IV.

1) Lloyd–Max Quantization: A scalar quantizer may be designed using the K-means algorithm described in Section II-C. However, because in one dimension the cells are restricted to be adjacent line segments, as shown in Fig. 4, one can instead use the well-known *Lloyd–Max quantizer*[8] [82], [91]. Given a pdf $p(x)$ and a number of levels L, this quantizer determines the intervals C_i and the reconstruction values y_i that minimize the average mse. The necessary conditions for the minimum are obtained by straightforward differentiation with respect to y_i and the interval boundary values g_i. The necessary conditions for optimality can be shown to be

$$g_i = \tfrac{1}{2}(y_i + y_{i-1}), \qquad 2 \leq i \leq L \tag{78}$$
$$g_1 = -\infty, \qquad g_{L+1} = \infty$$
$$y_i = \mathrm{cent}(C_i), \qquad 1 \leq i \leq L \tag{79}$$

where the centroid of C_i is simply the mean value of x in that interval. Equation (78) states that the interval boundaries must lie halfway between the reconstruction values. Note that (78) corresponds to (25) in the general case and (79) is the same as (27). For $L > 3$, (78) and (79) are solved itera-

[8]Also known as the Lloyd quantizer or the Max quantizer.

Exhibit Y Page 472

tively to obtain a set of optimal values. Those values may indicate only a local optimum. For the case when $p(x)$ is log-concave, the above necessary conditions are also sufficient for optimality [40] and the optimum will be global. In general, the optimum quantizer will not be uniform, i.e., the interval lengths will not be the same.[9] The interval spacing tends to be smaller where the pdf is larger. Performance, typically, must be evaluated by numerical methods, except when the number of quantization levels is large there is the asymptotic formula given by Algazi [4] for the rth-power distortion

$$D_r = \frac{2^{-r}}{r+1} L^{-r}\left[\int_{-\infty}^{\infty} [p(x)]^{1/(r+1)}\, dx\right]^{r+1} \qquad \text{(Lloyd–Max)}. \tag{80}$$

For $R = \log_2 L$ and $r = 2$, (80) can be written in decibels as

$$\mathscr{D}_2 = 10 \log_{10} D_2 = -6.02R + F_{LM}(p) \quad \text{dB} \tag{81}$$

where script $\mathscr{D}_2$ is the distortion in decibels and $F_{LM}(p)$ is a constant that depends on the pdf shape. Note again the -6.02-dB/bit behavior for large R.

Fig. 11 shows the normalized mse obtained by the Lloyd–Max quantizer as a function of the rate $R = \log_2 L$ for four pdfs. While the Gamma pdf has the lowest $D(R)$ of

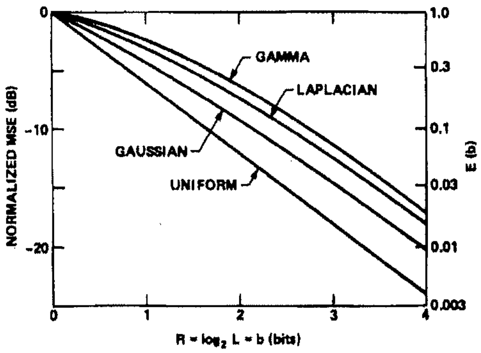

Fig. 11. Normalized mse for four memoryless sources using Lloyd–Max quantization ($R = \log_2 L$), plotted from Jayant and Noll [74, Table 4.4].

the four pdfs, it results in the highest distortion when using the Lloyd–Max quantizer. Columns 6 and 7 of Table 1 show the asymptotic difference between the Lloyd–Max quantizer and the distortion-rate function. These differences are the maximum that can be gained potentially from vector quantization.

2) Constrained-Entropy Quantization: In Lloyd–Max quantization, the number of levels L is fixed and the bit rate is defined simply as $\log_2 L$. However, if the reconstruction values y_i are not equally probable, we can use entropy coding to reduce the rate below $\log_2 L$ to $H(y)$. One can go even further and restate the optimization problem to minimize the distortion *subject to a given entropy $H(y) = R$.* We shall call the resulting quantizer a *constrained-entropy quantizer.* The number of levels L can now be any desired value including infinity. Gish and Pierce [51] have shown that, *at high bit rates, the uniform quantizer is the optimum constrained-entropy quantizer.* With very mild restrictions, this result is true for all pdfs and distortion measures. More recently, Farvardin and Modestino [33] showed experimentally that, with mse distortion, the uniform quantizer is very close to optimal even at very low rates. The obvious conclusion is that, if one is willing to perform variable-rate entropy coding, then the uniform quantizer is always the scalar quantizer of choice.

[9]Note that a uniform quantizer is one whose interval lengths are equal. However, depending on the pdf shape and the distortion measure, the output levels of a minimum-distortion uniform quantizer will not be equally spaced, in general.

The asymptotic (high bit rate) performance for the constrained-entropy quantizer is given in [51], and can be written for an rth-power distortion measure as

$$D_r = \frac{2^{-r}}{r+1} 2^{rh(x)} 2^{-rH_{\min}} \quad \text{(constrained entropy)} \tag{82}$$

where $H_{\min}$ is the entropy of the constrained-entropy quantizer outputs and $h(x)$ is the differential entropy of the source. Equation (82) can be written in decibels for $r = 2$

$$\mathscr{D}_2 = -6.02 H_{\min} + F_{CE}(p) \quad \text{dB} \tag{83}$$

where F_{CE} is another constant that depends on the pdf shape.

It is interesting that, for the mse distortion, there is a simple relationship between $\log_2 L$ of the Lloyd–Max quantizer and $H_{\min}$. In particular, we have from [51] that, for large L

$$H_{LM} - H_{\min} = \tfrac{1}{3}(\log_2 L - H_{\min}) \tag{84}$$

where H_{LM} is the entropy of the Lloyd–Max quantizer outputs. This relation shows the additional benefit of constrained-entropy quantization over entropy coding the outputs of the Lloyd–Max quantizer.

Another significant result obtained by Gish and Pierce [51] is that the constrained-entropy quantizer approaches the rate-distortion bound within a fixed constant that is dependent only on the distortion measure and not on the pdf. This constant, derived for an rth-power distortion measure, is given by

$$H_{\min} - R(D_r) = \frac{1}{r} \log_2 \frac{re}{1+r} + \log_2 \Gamma\left(1 + \frac{1}{r}\right) \tag{85}$$

where Γ is the Gamma (factorial) function. For the mse distortion, (85) reduces to

$$H_{\min} - R(D_2) = \frac{1}{2} \log_2 \frac{\pi e}{6} = 0.255 \text{ bits}. \tag{86}$$

Fig. 12 shows a plot of (85) as a function of r. The horizontal scale has been chosen to illustrate the important result that for $r = \infty$, which corresponds to the minimax criterion, the curve goes to zero, and $H_{\min} = R(D_\infty)$. Therefore, *a uniform scalar quantizer with entropy coding can achieve the rate-distortion function for the minimax criterion.* For $r = 1$, the constrained-entropy quantizer can approach $R(D)$ within 0.443 bits, and within 0.255 bits for $r = 2$.

The low-rate region ($R < 3$ bits) has been explored by a number of researchers (see [74]), with the results of Farvardin and Modestino [33] being the most useful for our purposes. For the mse distortion, the constrained-entropy quantizer at low rates is even closer to $D(R)$ than 0.255 bits, especially for the more peaked pdfs such as the Laplacian and the Gamma densities. In fact, of all the nonuniform pdfs tested, the Gaussian pdf registers the least gain and

Exhibit Y Page 473

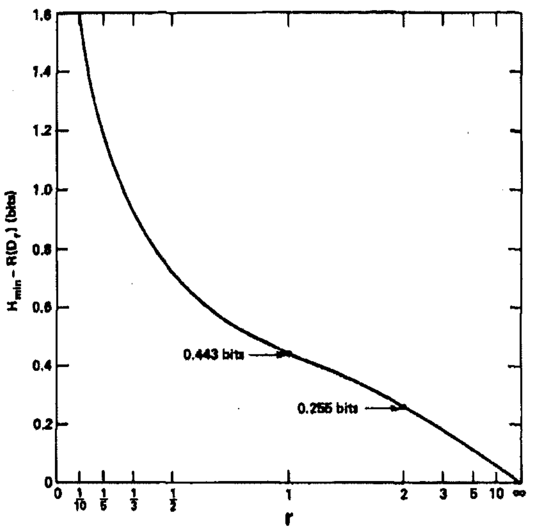

Fig. 12. The difference between the entropy of a constrained-entropy (uniform) quantizer and the rate-distortion function at high bit rates for an rth-power distortion function. This difference is independent of the pdf of the memoryless source; it depends only on r. For $r = \infty$, the minimax distortion, that difference is zero. The data for the plot were taken from Gish and Pierce [51].

lies the furthest away from its $D(R)$ function at low rates. Figs. 13 and 14 show plots of the mse distortion with the constrained-entropy quantizer (solid curves) compared to Lloyd–Max quantization and the distortion-rate functions for the Gaussian (Fig. 13) and the Laplacian (Fig. 14) memoryless sources. The constrained-entropy curve for the

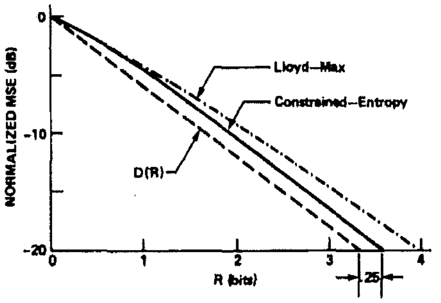

Fig. 13. Constrained-entropy quantization for a memoryless Gaussian source, compared to Lloyd–Max quantization and the distortion-rate function. (From Farvardin and Modestino [33].)

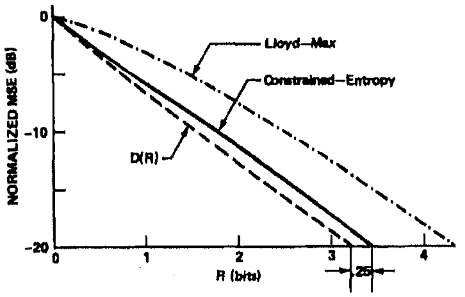

Fig. 14. Same as Fig. 13 for a memoryless Laplacian source. (From Farvardin and Modestino [33].)

Laplacian pdf is much closer to its distortion-rate function than the Gaussian curve is to its $D(R)$, while the opposite is true for the Lloyd–Max curves.

We saw above how constrained-entropy quantization (with entropy coding) takes advantage of the pdf shape to minimize the bit rate at a given distortion. The difference between the solid curves and the $D(R)$ curves in Figs. 13 and 14 is what is not achievable by scalar quantization. Vector quantization with very large N is capable of closing that gap without the need for entropy coding. At the low data rate of $R = 1$ bit in Fig. 13, being able to reach the distortion-rate bound for a Gaussian source would mean a reduction of about 25 percent in bit rate over the scalar quantizers. This result will be important for the coding of the speech prediction residual since it is modeled well as a memoryless Gaussian source.

C. Asymptotic Vector Quantization

We now present some of the known theoretical results for vector quantization. Most of the results below were derived under asymptotic conditions: either L is large (i.e., small distortion) and the vector size N is arbitrary, or N is large *and* L *is arbitrary*. We begin by discussing the optimum L-level quantizer in N dimensions, which is analogous to Lloyd–Max quantization in the scalar case, followed by constrained-entropy quantization in the vector case. It is important to notice that, unlike the results of the previous section which applied only to memoryless sources, *the VQ results below apply to sources with arbitrary pdfs, including linear and nonlinear dependencies*, unless noted otherwise.

The asymptotic formulas for scalar quantizer performance mentioned in Section III-B have analogues for vector quantizers. Zador [141] showed that, for large L, the optimum (minimum rth-power distortion) L-level quantizer in N dimensions has a distortion

$$D_r = A(r, N) L^{-r/N} \left[\int_{-\infty}^{\infty} [p(\mathbf{x})]^{N/(N+r)} \, d\mathbf{x} \right]^{(N+r)/N} \tag{87}$$

where $p(\mathbf{x})$ is the N-dimensional pdf of the vector process $\mathbf{x}$, and $A(r, N)$ is a term that is independent of the pdf; it represents how well cells can be packed in N-dimensional space for the rth-power distortion measure. Equation (87) is analogous to (80) in the scalar case, but it is important to note that (80) was derived for a memoryless source while (87) is true for an arbitrary source. For the mse distortion, (87) may be written in decibels as

$$\begin{aligned} \mathscr{D}_2 &= -\frac{6.02}{N} B + F_{VQ}(p, N) \quad \text{dB} \\ &= -6.02R + F_{VQ}(p, N) \quad \text{dB} \end{aligned} \tag{88}$$

where $B = \log_2 L$ is the number of bits per vector, R is the number of bits per dimension, and $F_{VQ}(p, N)$ is a term that depends on the pdf $p(\mathbf{x})$ and the number of dimensions.

The difficulty in utilizing the above expressions is that $A(r, N)$ is known exactly for very few cases. For other than $N = 1$, only $A(2, 2)$ is known exactly. For two dimensions and the mse criterion [35], [47], optimal vector quantizer performance (without entropy coding) is obtained with hexagonal regions, for large L. Note that, except for a uniform pdf, these hexagons will not be regular and will

Exhibit Y Page 474

not have the same size in general. For higher dimensions, there are a number of upper and lower bounds on $A(r, N)$, but such bounds, though theoretically interesting, appear to be of little practical value. Recently, Conway and Sloane [23] conjectured a lower bound on $A(2, N)$ that seems interesting and useful (see Fig. 15). Other useful information on performance bounds is contained in [47], [139].

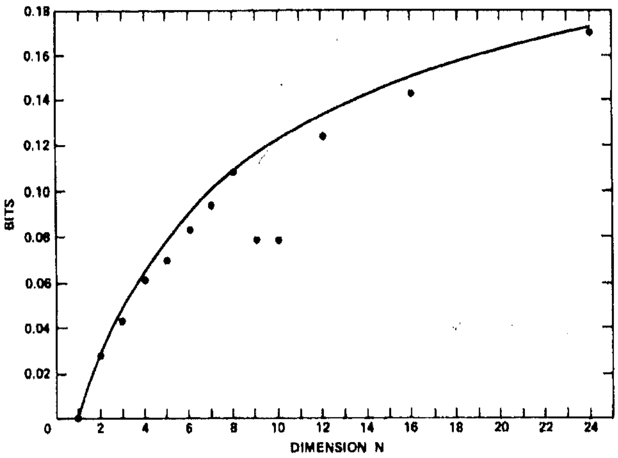

Fig. 15. The savings in bits when using certain N-dimensional lattice quantizers compared to quantization with N-dimensional cubes for a uniform pdf and high bit rate. The data were computed from Conway and Sloane [23, Table I] as follows:

$$\text{BITS} = [10 \log_{10} 0.08333/G_n]/6.02$$

where G_n is as defined in [23]. The solid curve is a conjectured bound given by Conway and Sloane [23] and the dots are the savings for the best known lattice quantizer for each N.

Although the optimal N-dimensional quantizer and its performance are not known in most cases, some understanding has been obtained about its structure. Zador [141] has shown that the optimum density of output values to use for random quantizer selection is proportional to $p^{N/(N+r)}$, where p is the pdf of the vector $\boldsymbol{x}$. (A related result is also obtained by Gersho [47].) For very large N ($N \gg r$), therefore, the output values should have a density similar to $p(\boldsymbol{x})$ for optimum performance.

In a manner similar to scalar quantizer design, one can constrain the entropy rather than the number of levels for the vector quantizer; we call the resulting quantizer the *constrained-entropy vector quantizer*. For such a quantizer, Zador [141] gives the asymptotic rth-power distortion as

$$D_r = B(r, N) 2^{rh(\boldsymbol{x})/N} 2^{-rH_{\min}/N} \tag{89}$$

where $B(r, N)$ is a term similar to $A(r, N)$ in (87). Equation (89) is analogous to (82) in the scalar case. One could write (89) for $r = 2$ to show the -6.02-dB/bit behavior.

Again, values of $B(r, N)$ are known for only a few cases. However, if entropy coding is to be used, there are suboptimal solutions for which the performance is known for large L. One solution, considered by Gish and Pierce [51], is analogous to the solution in the scalar case; namely, apply a uniform scalar quantizer to each of the vector dimensions and perform *joint entropy coding* on the vector outputs. The quantization interval should be the same in all dimensions, which is equivalent to specifying an N-dimensional cube for the cells of the vector quantizer. The result for this coding procedure is that the bit rate will differ from the rate-distortion function by an amount equal to that shown in Fig. 12 for an rth-power distortion measure, independent of the shape of the multidimensional pdf. For the mse, this difference is again 0.255 bits per dimension. This result is true at high bit rates; experiments similar to those performed by Farvardin and Modestino [33] for the scalar case will need to be done for the vector case to test the performance at low bit rates. The 0.255-bit differential that exists for the mse criterion when quantizing with cubes can be reduced by using different cell shapes for different dimensions, as we shall see below in the discussion under space packing.

While the VQ method outlined above is very simple indeed and requires a minimum amount of design and computations, it requires a very large amount of storage. Note that the entropy coding needs to be performed in N-dimensional space. Now, when we use entropy coding we typically use a large number of levels per dimension, therefore, the total number of levels in N-dimensional space will be very large, and the entropy coding will require a storage location for each of the codewords. In essence, we are substituting storage for computation and the amount of storage needed may be excessive for many applications.

Another important result from rate-distortion theory is that, for fixed L and for very large N, the entropy of the vector quantizer approaches $\log_2 L$ [123]. This implies that the output values y_i must have equal probability. (Note that the output values may have a *density* proportional to $p(\boldsymbol{x})$ but still have *discrete* probabilities that are equal.) This result is satisfying in that it confirms the fact that a vector quantizer can achieve its optimal performance for high dimensions without the need for entropy coding. In other words, the fixed-L and constrained-entropy quantizers achieve the same performance for high dimensions. This statement is in sharp contrast to the scalar case ($N = 1$) where the two types of quantizers give different performance.

Space packing: An interesting set of results have been obtained in studying the benefits of packing N-dimensional space with different types of polytopes (volumes bounded by planes) for the mse criterion. In fact, in a manner similar to the hexagon result mentioned above for $N = 2$, it has been conjectured by Gersho [47] that, asymptotically for large L and for *any pdf*, the optimum fixed-L vector quantizer partitions the space into cells whose shapes are derived from a specific polytope for each N.

A special set of polytopes, known as parallelohedra, have been studied by Gersho [47] and by Conway and Sloane [21] for the purpose of quantizing a uniform pdf in N dimensions. The centers of these parallelohedra lie on a *lattice* in N-dimensional space.[10] The lattice points are the output values of the quantizer (i.e., they constitute the codebook), and they form a regularly spaced array of points in N-dimensional space. Such a quantizer is termed a *lattice quantizer*. One very attractive feature of lattice quantizers is

[10]The concept of a lattice is of importance in a number of diverse areas, including the geometry of numbers [19], solid-state physics [143], and image processing [29].

Exhibit 4 Page 475

that, given an input vector $\boldsymbol{x}$, it is relatively straightforward to compute the nearest lattice output value [22], without having to compare $\boldsymbol{x}$ against all codebook values and without having to store all codebook values. Conway and Sloane [23] have investigated the use of a number of lattices in the vector quantization of a uniform pdf for large L and using the mse criterion. Fig. 15 shows the saving in bits when those lattices are used relative to using cubes as quantization regions. For each dimension, the lattice used is different. For $N = 1$ we have, of course, the uniform cube quantizer. For $N = 2$, it is the hexagonal lattice, and for $N = 3$, the polyhedron is known as the truncated octahedron. Note that the saving in bit rate increases with N for the different lattices except for $N = 9$ and $N = 10$ for which no lattices have been found that perform better than $N = 8$. The value of 0.028 bits for $N = 2$ is the same as that obtained in Example 3 earlier.

Although the results in Fig. 15 were obtained for a uniform pdf, they have wider applicability. One can show that, if entropy coding is used, then for *any pdf*, the lattice quantizers with large L achieve an entropy less than the cube quantizer by an amount equal to the rates shown in Fig. 15. Since from the results of Gish and Pierce the bit rate of the cube quantizer is 0.255 bits above the rate-distortion function, the bit rate for the lattice quantizers is even closer to $R(D)$ by an amount equal to that shown in Fig. 15 for different dimensions. For example, for $N = 8$, the lattice quantizer with entropy coding will have a bit rate that is only $0.255 - 0.109 = 0.146$ bits greater than $R(D)$ for large L. However, because of the potentially excessive amount of storage needed for the codewords, this method may not be practical. In the remainder of the paper we shall discuss methods that do not require entropy coding.

Experimental results: Much of this section has been devoted to theoretical asymptotic vector quantization results. Few attempts have been made to see how these results may apply in practice and for relatively small values of L and N. For a memoryless Gaussian source, the best VQ results with $R = 1$ bit/sample show practically no improvement over scalar quantization for $N = 2$, a reduction in mse of 0.09 dB (about 0.015 bits) for $N = 3$, and a reduction of about 0.52 dB (about 0.087 bits) for $N = 6$ [36], [58], [81], which is a good improvement for $R = 1$ bit. These results were obtained with no entropy coding.

Much greater gains are achievable for certain non-Gaussian sources. For example, for a memoryless Gamma source and $R = 1$ bit/sample, Fischer and Dicharry [36] obtained a substantial reduction in mse for $N = 6$. However, the resulting mse was still approximately 2 dB higher than $D(R)$ and well above the distortion of the constrained-entropy uniform quantizer, which was found by Granzow and Noll [55] to be only 0.7 dB above $D(R)$. However, this result serves to illustrate that, for cases where Lloyd–Max quantization performs poorly relative to $D(R)$, and entropy coding is not desired or cannot be used, one can make good use of vector quantization to improve performance substantially.

Results with a single-pole Gaussian source have also been obtained at $R = 1$ bit/sample, which show that VQ could outperform differential PCM (DPCM) coding with vector lengths $N > 3$ [60]. But, for $N = 7$, the VQ mse was still 1.6 dB higher than $D(R)$.

IV. Scalar versus Vector Quantization of Vector Sources

We indicated in Section II that the redundancy owing to linear dependency (correlation) can be utilized effectively by vector as well as scalar quantization, provided the latter is performed after the appropriate coordinate transformation. In this section we show how best to use scalar quantizers in the quantization of vector components. The purpose for this presentation is twofold. We wish to give the reader tools to help assess how much of the vector quantization performance in a particular application takes advantage of linear dependencies, and to help evaluate whether a scalar or a vector quantizer is more cost effective in that application, given other real-world constraints.

Throughout this section we use the mse as the distortion measure. The first two subsections are devoted to the scalar quantization of vector sources and the last subsection compares scalar to vector quantization. First, we show that, given a total number of bits to allocate to scalar coding of the various vector components, it is best (minimum mse) to allocate different number of bits to components with different variances. A bit-allocation procedure is described which yields the minimum mse. We then show that, if the vector components are correlated, one can reduce the mse further by first performing a decorrelating transformation or rotation on the vector, followed by independent (scalar) quantization of the components of the rotated vector, utilizing the bit-allocation procedure. Finally, we compare the performance of scalar and vector quantizers in a particular application.

The subject matter of this section is very applicable to the coding of linear prediction parameters (such as log-area ratios), which are generally correlated and have different probability distributions and different variances (see Section IV-C). It is also applicable to the transform coding of speech and other waveforms.

A. Bit Allocation

We are given a random vector $\boldsymbol{x}$ whose components x_k have identical pdf shapes but generally unequal variances σ_k^2. (The case of different pdf shapes will be treated below.) We wish to allocate a given number of bits B among the components and use scalar quantizers to quantize each component x_k independently, in such a way as to minimize the average vector distortion. Let R_k be the number of bits allocated to component x_k and let the distortion in quantizing that component be

$$D_k = \mathscr{E}(x_k - y_k)^2 \tag{90}$$

where y_k is the quantized value of x_k. The problem then is to determine the set $\{R_k\}$ so as to minimize the average distortion

$$D = \frac{1}{N}\sum_{k=1}^{N} D_k \tag{91}$$

subject to

$$\frac{1}{N}\sum_{k=1}^{N} R_k = \frac{B}{N} = R \tag{92}$$

where R is the average bit rate per component.

Exhibit Y Page 476

Let us assume for the present that we are operating in the high bit-rate region of the quantization distortion curve for each of the components (usually $R > 3$). Then we can write from the results of Section III-B

$$R_k = \rho + \frac{1}{2}\log_2 \frac{\sigma_k^2}{D_k} \tag{93}$$

where ρ is a constant value that depends on the pdf shape and on the particular scalar quantization method used. (For example, for a Lloyd–Max quantizer, we see from Fig. 11 that $\rho = 0$ for a uniform pdf and $\rho = 0.722$ for a Gaussian pdf.) From (91) and (93) we have

$$D = \frac{1}{N}\sum_{k=1}^{N} \sigma_k^2 2^{-2(R_k-\rho)}. \tag{94}$$

(Note that D in (94) has a slope of $-6.02/N$ decibels/bit, as expected for high bit rates.) The problem now is to find the R_k values that minimize D in (94) subject to (92). Using Lagrange multipliers, one can show that the distortions D_k must be equal

$$D_{\min} = D_k = 2^{-2(R-\rho)}\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}, \quad \text{for all } k \tag{95}$$

and the bit allocation is given by

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}}. \tag{96}$$

If the components have different variances, the result implies that the number of bits used to quantize each component will be different.

By allocating a different number of bits to different components we realize a lower distortion than we would, had we used an equal number of bits R to quantize each component. In the latter case the total distortion would be

$$D_0 = 2^{-2(R-\rho)}\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2. \tag{97}$$

Therefore, the ratio of the minimum distortion $D_{\min}$ to the equal allocation distortion D_0 is

$$\frac{D_{\min}}{D_0} = \frac{\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}}{\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2} = \frac{\mathrm{GM}\{\sigma_k^2\}}{\mathrm{AM}\{\sigma_k^2\}} = \gamma \leq 1 \tag{98}$$

which is the ratio of the geometric mean to the arithmetic mean of the component variances, with equality iff all variances are equal. Thus with bit allocation, one would have an increase in SNR of $-10\log_{10}\gamma$ decibels. (Note that (98) was derived assuming the same pdf and high bit rates for all components.)

If $\gamma \ll 1$, implying that the component variances have a large dynamic range, then the bit-allocation scheme in (96) would result in a wide range of bit rates R_k. Some of these may be very low (even though the average may be high) and some may even be negative. Low rates would violate our assumption of high rates that allowed us to write (93), and negative rates are, of course, unallowable. The latter condition results for components whose variances are less than $D_{\min}$ in (95). For such cases, one would not transmit anything because by simply using the mean value at the receiver one would have at worst a distortion equal to the signal variance. If such negative bit rates are obtained from the bit-allocation scheme, the corresponding components are allocated zero bits. When negative values of R_k are thus increased to zero, the rates of other components will have to be modified to maintain the same average rate R. Several procedures have been developed which reoptimize the allocation. Segall [120] has solved the above optimal *continuous* bit allocation problem, assuming independent Gaussian components. Fox [43] had earlier presented an algorithm for optimal *integer* bit allocation in the general case where the pdfs of the components may be different. Below we give a brief description of the algorithm and the conditions for its applicability.

Optimum Integer Bit Allocation: Let us assume that we wish to have a bit allocation where all rates R_k are integers. We associate with each component x_k a normalized (unit-variance) quantization distortion function $E_k(b)$, which gives the distortion as a function of the number of bits b, assuming x_k has unit variance. (We assume that each x_k may have a different pdf.) Clearly then, the distortion of x_k is given by

$$D_k(b) = \sigma_k^2 E_k(b). \tag{99}$$

$E_k(b)$ is an actual curve obtained by applying a specific quantizer to the random variable x_k with its specific pdf. The quantizer can be a Lloyd–Max quantizer or a constrained-entropy quantizer or any other quantizer of interest. The main thing is to generate the curve $E_k(b)$ for each component. Of course, if $E_k(b)$ is not optimized in any way to the component x_k, then the results may be suboptimal. (Fig. 11 shows $E(b)$ for a Lloyd–Max quantizer for four different pdfs.)

The bit-allocation procedure simply assigns the next bit to the component that causes the maximum incremental decrease in the overall distortion. The procedure is as follows:

Step 1: For each bit rate b, calculate the incremental decrease in the distortion when a bit is added to each component:

$$\Delta_k(b) = \sigma_k^2[E_k(b-1) - E_k(b)], \quad 1 \leq k \leq N, \quad b = 1, 2, \cdots. \tag{100}$$

Step 2: Sort the values $\Delta_k(b)$ in decreasing order.

Step 3: Assign the given bits one by one according to the resulting order.

(In actual implementation, the values $\Delta_k(b)$ are computed sequentially as they are needed in allocating the bits.) This algorithm can also be applied to assigning an integral number of *levels* to each component. The variable b would then refer to the number of levels.

The above algorithm gives the optimal solution when all quantization distortion functions $E_k(b)$ are monotone decreasing and convex, i.e., successive bits decrease the quantization error by an amount smaller than earlier bits allocated to that component. For the general case, where the quantization distortion functions are nonconvex or even

Exhibit Y Page 477